IN-62
175543
P.200

NASA Contractor Report 4516

Advanced Launch System Multi-Path Redundant Avionics Architecture Analysis and Characterization

Robert L. Baker

CONTRACT NAS1-17964
JUNE 1993

NASA

NASA Contractor Report 4516

501147

Advanced Launch System Multi-Path Redundant Avionics Architecture Analysis and Characterization

Robert L. Baker
Research Triangle Institute
Research Triangle Park, North Carolina

Prepared for
Langley Research Center
under Contract NAS1-17964

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1993

Contents

List of Figures

List of Tables

Acronyms, Abbreviations and Symbology

ADAS	Architecture Design and Assessment System
ADP	Advanced Development Program
AGN&C	Adaptive Guidance, Navigation and Control
AIPS	Advanced Information Processing System
ALS	Advanced Launch System
ASSIST	Tool for Specifying Reliability Models
BIT	Built-in-Test
BIU	Bus Interface Unit
BM/C^3	Battle Management Communications Command and Control
CAP-32	32 bit Common Avionics Processor
CP	Computational Processor
DIU	Device Interface Unit
DOF	Degrees of Freedom
DPM	Dual Port Memory
EEPROM	Electronically Erasable Programmable Read Only Memory
EME	Electromagnetic Environment
FDIR	Fault Detection Isolation and Recovery
FLV	Future Launch Vehicle
FTM	Fault Tolerance Module
FTP	Fault Tolerant Processor
GN&C	Guidance, Navigation and Control
GPS	Global Positioning System
G&N	Guidance and Navigation
HERF	High Energy Radio Frequency
HSDB	High Speed Data Bus
HW	Hardware
IC	Intercomputer Network
ICIS	Intercomputer Interface Sequencer
ID	Identification
IMU	Inertial Measurement Unit
IOS	Input/Output Sequencer
ISA/RTL	Instruction Set Architecture/Register Transfer Language
JIAWG	Joint Integrated Avionics Working Group
LCC	Life Cycle Cost

LDL	Local Data Link
MIL-HDBK	Military Handbook
MIPS	Millions of Instructions Per Second
MPRAS	Multipath Redundant Avionics Suite
MTBF	Mean Time Between Failure
OS	Operating System
P/A	Propulsion/Avionics
PAVE PILLAR	Air Force Program to Develop Common Avionics Computing Modules
PDR	Preliminary Design Review
PE	Processing Elements
PIbus	Processor Interface Bus for JIAWG Modules
PROM	Programmable Read Only Memory
QMR	Quadruplex Modular Redundancy
RADC-TR	Rome Air Development Center Technical Report
RAM	Random Access Memory
RB:BB	Receiver Bandwidth to Bus Bandwidth Ratio
RCS	Reaction Control System
RDI	Remote Data Interface
ROM	Read Only Memory
RTI	Research Triangle Institute
SCP	Self Checking Pair
SDIO	Strategic Defense Initiative Office
SIM	Sensor Input Module
SRD	System Requirements Document
SRR	System Requirements Review
STM	System Test and Maintenance Modules
SURE	Tool for Analyzing Unreliability Based on Semi-Markov Models
TDPA	Table Driven Proportional Access
TLM	Telemetry
TMbus	Test and Maintenance Bus for JIAWG Modules
TT&C	Telemetry, Tracking and Command
TVC	Thrust Vector Control
UTC	Universal Coordinated Time
VHDL	VHSIC Hardware Description Language
VHM	Vehicle Health Monitor
VLSI	Very Large Scale Integration

1. INTRODUCTION

1.1. Purpose

The objective of the Multi-Path Redundant Avionics Suite (MPRAS) program is the development of a set of avionics architectural modules which will be applicable to the family of launch vehicles required to support the Advanced Launch System (ALS). To enable ALS cost/performance requirements to be met, the MPRAS must support autonomy, maintenance and testability capabilities which exceed those present in conventional launch vehicles. The multi-path redundant or fault tolerance characteristics of the MPRAS are necessary to meet avionics reliability requirements.

A complex, real-time distributed computing system is needed to meet the ALS avionics system requirements. General Dynamics, Boeing Aerospace and C.S. Draper Laboratory have proposed system architectures as candidates for the ALS MPRAS. The purpose of this document is to report the results of independent performance and reliability characterization and assessment analyses of each proposed candidate architecture and qualitative assessments of testability, maintainability and fault tolerance mechanisms. These independent analyses were conducted as part of the MPRAS Part 2 program and were carried under NASA Langley Research Contract NAS1-17964, Task Assignment 28.

The characterization and assessment analyses were directed toward identifying strengths, weaknesses, limitations and development risks for each architecture. At the outset of this effort, an evaluation plan was developed which called for comparative evaluations of the architectures against a common baseline set of application requirements. As the program evolved, each architecture was designed to meet different detailed application requirements as defined by the developer. While there was considerable similarity between these requirements, there were instances where the requirements differed radically. Consequently, comparative evaluations were not possible in certain areas.

1.2. General System Description

Figure 1.1 is a generic diagram showing the major elements of a digital avionics system appropriate for the MPRAS application. The system is composed of: 1) sensors that provide measurements of physical parameters which are necessary to implement the desired control functions, 2) a sensor communications network to deliver sensor data to the distributed computing resources, 3) distributed computing resources to carry out the control computations, 4) a computer communications network to provide for data and control communications between computing resources, 5) operating system and application software which implement the desired avionics functions, 6) an actuator data distribution network which delivers control information to the actuators, 7) the actuators that effect the desired control actions, and 8) a power system that provides the necessary electrical power to all elements and interfaces to other vehicle or ground support subsystems.

The major functions implemented by the digital avionics system for core and booster components include:

1. engine or propulsion control,
2. adaptive guidance navigation and flight control,
3. fluids management,
4. integrated health monitoring,
5. power management,
6. data recording,
7. communications and telemetry, and
8. mission control.

1.3. Architecture Configurations and Coarse Module Breakdowns

In order to determine rough complexity characteristics for the MPRAS application, modules of the candidate architectures were configured to implement a hypothetical case. This case was derived to be generally within and representative of the MPRAS requirements as represented by the Boeing Aerospace MPRAS System Requirements Document and the General Dynamics MPRAS Point Design Evaluation Report. It does not, however, represent a specific application. This effort was undertaken to determine the overall complexity of the application in terms of module counts and to determine each architecture sensitivity to application complexity.

A hypothetical launch vehicle with three engines in the core stage and five engines in the boost stage was considered. For each engine the propulsion control used 30 parameter measurements that must be updated at a 100 Hz rate. Of the 30, 15 used dual redundant sensors and 15 require single sensors. A total of 45, 100Hz, sensors

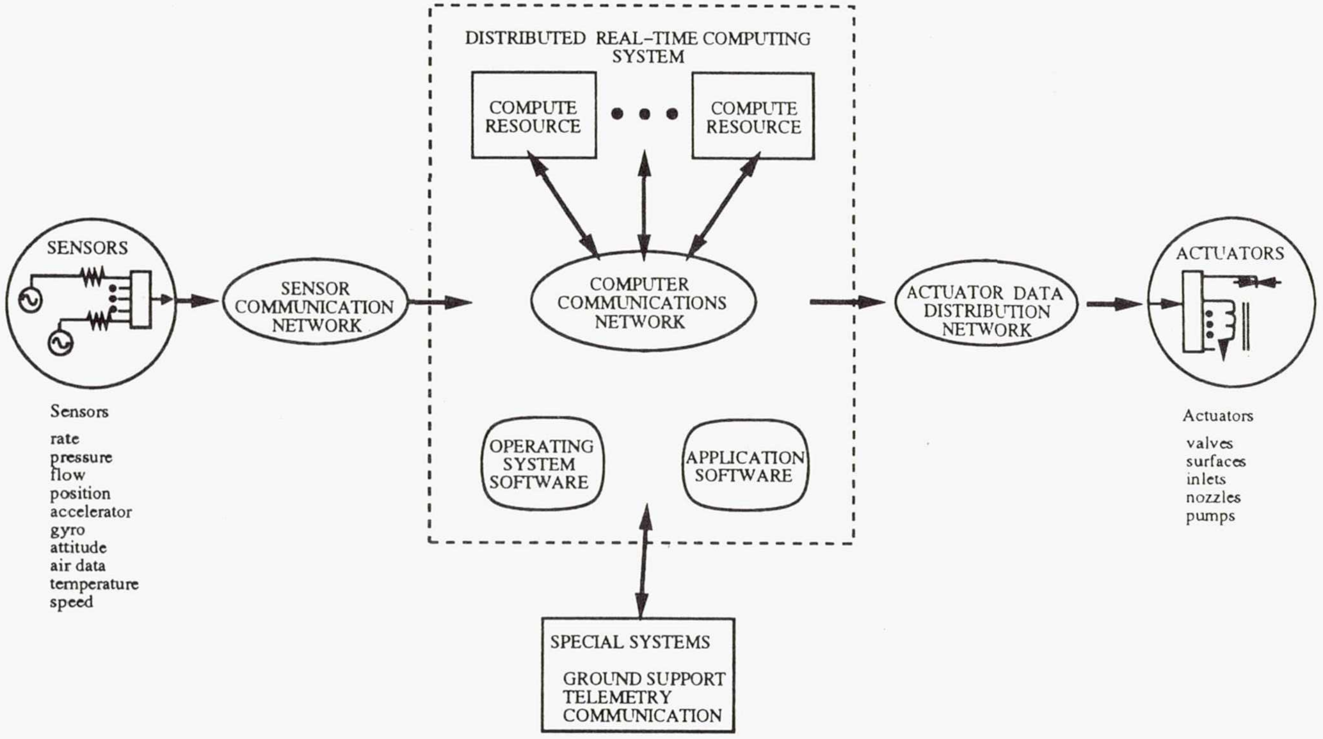

SENSORS
Sensors
rate
pressure
flow
position
accelerator
gyro
attitude
air data
temperature
speed
SENSOR COMMUNICATION NETWORK
DISTRIBUTED REAL-TIME COMPUTING SYSTEM
COMPUTE RESOURCE
COMPUTE RESOURCE
COMPUTER COMMUNICATIONS NETWORK
OPERATING SYSTEM SOFTWARE
APPLICATION SOFTWARE
SPECIAL SYSTEMS
GROUND SUPPORT
TELEMETRY
COMMUNICATION
ACTUATOR DATA DISTRIBUTION NETWORK
ACTUATORS
Actuators
valves
surfaces
inlets
nozzles
pumps

Figure 1.1.

were used for propulsion. In addition, 30 parameter measurements, updated at a 1 Hz rate were used. Similarly, 15 used dual redundant sensors and 15 used single sensors for a total of 45 additional sensors. Finally, two dual redundant acoustic vibrator sensors sampled at a 2 KHz rate were used. Actuators required for each engine were 16 updated at a 1 Hz rate and 4 updated at a 50 Hz rate. To support fluids management, adaptive guidance, navigation and flight control an additional 540 sensors and 240 actuators were distributed within the core and booster. A block of inertial measurement sensors and a GPS receiver and processor were assigned to provide essential inputs for the navigation function. A total of 1290 sensors and 400 actuators were required for this example. This number is well within the estimated 3000 to 6000 sensor/actuator requirements set for MPRAS. Functions such as integrated health monitoring and data recording were not considered.

Except where noted, modules represent a circuit card. The modules that make up each architecture were configured to meet the requirements of this case. No claim is made that the configurations are optimum.

Following General Dynamics guidelines, a triplex system with a remote data interface (RDI) dedicated to each engine is used. Additional, RDI's are used to acquire sensor data from sensor input modules (SIM) distributed throughout the vehicle. The SIM was assumed to handle 15 sensor signals with necessary excitation signal conditioning, analog/digital conversion and testing capability. Since this module has not been completely defined and since it may require more than one circuit card to realize, module counts for this function are not directly comparable to other module counts. The processing required to carry out MPRAS functionality is provided by the Inertial Measurement Unit (IMU) and the Vehicle Management processor (VMP). The proposed General Dynamics self-checking processor/memory module is substantially more complex than the processor or memory modules used in the other architectures and may require more than one circuit card for implementation. Consequently, module counts shown for the General Dynamics configuration may tend to be less than that actually required. This configuration does not incorporate the more distributed sensor data collection scheme currently being considered by General Dynamics under MPRAS Part III.

The configuration for the Boeing modules following Boeing's guidelines is quadruplex and assigns only one processing site in each stage to propulsion control. Processors are assigned to perform mission processing (MP) and guidance, navigation, and control (GNC). Note that the computing requirements for propulsion control, if they are as high as 5-10 MIPS/engine as some estimates indicate, may require more processors to control the eight engines. Interface hardware for the 1773a and HSDB buses are required for the processing sites and I/O processors. Module requirements for the local signal conditioner unit (LSC) which was not fully defined by Boeing assumes that six modules are required to interface to 30 sensors and/or actuators and that an additional two modules are required for A/D conversion, testing and transducer bus interfaces.

The AIPS module count is based on a quadruplex system and on assigning one fault-tolerant processor (FTP) to each engine. Additional processors are assigned to perform mission processing (MP) and guidance navigation and control (GNC)The module breakdown for this configuration follows the proof-of-concept implementation of AIPS. Consequently, the number of unique modules and hence the total module count will tend to be higher than module counts for a repackaged AIPS which takes advantage of projected microelectronics technology, and higher than module counts for the other architectures which are based on projected technology. Device interface units (DIU) provide for the acquisition of sensor data and the distribution of actuator control data. The DIU module counts assumes 15 inputs or outputs per module for signal conditioning and sensor interfacing. The intercomputer network (IC) is triplex redundant and the fault-tolerant I/O network is assumed to have at least dual redundancy. Network node hardware provides for circuit switching on the IC and I/O networks. Some I/O links can be used in AIPS to provide for additional reliability of the networks. However, the constraints of routing these spare links within the vehicle may render this AIPS feature unusable.

Figures 1.2, 1.3, 1.4, 1.5 show configurations for each architecture. Tables 1.1, 1.3,and 1.2 summarize module counts.

A direct comparison of module counts for each architecture is not appropriate given the assumptions going into their derivation. However, a number of general observations can be made from these coarse module counts. First, between 50% and 85% of the modules are directly involved in sensor/effector I/O. Second, if the module counts are divided by the assumed redundancy the number of modules in each redundant path is nominally 150 modules plus or minus 20%. This will dictate average module failure rates, which are more than two orders of magnitude lower than the necessary channel failure. Further, each channel will have nominally 50,000 circuit card electrical contacts. Third, the more distributed sensor data collection characteristics of the Boeing architecture, which also was assumed for the AIPS architecture, results in substantially more mechanical enclosures and hence more maintenance access points within the vehicle. Since General Dynamics has indicated a desire to use a more distributed sensor data collection scheme in their current MPRAS Part III activities, the significant difference between enclosure requirements for the architectures would be reduced.

A final observation regarding the total number of processors used in these modules is in order. Since the standard JIAWG modules such as the HSDB interface and the 1773 interface contain 1750A processors, the total number of processors used in

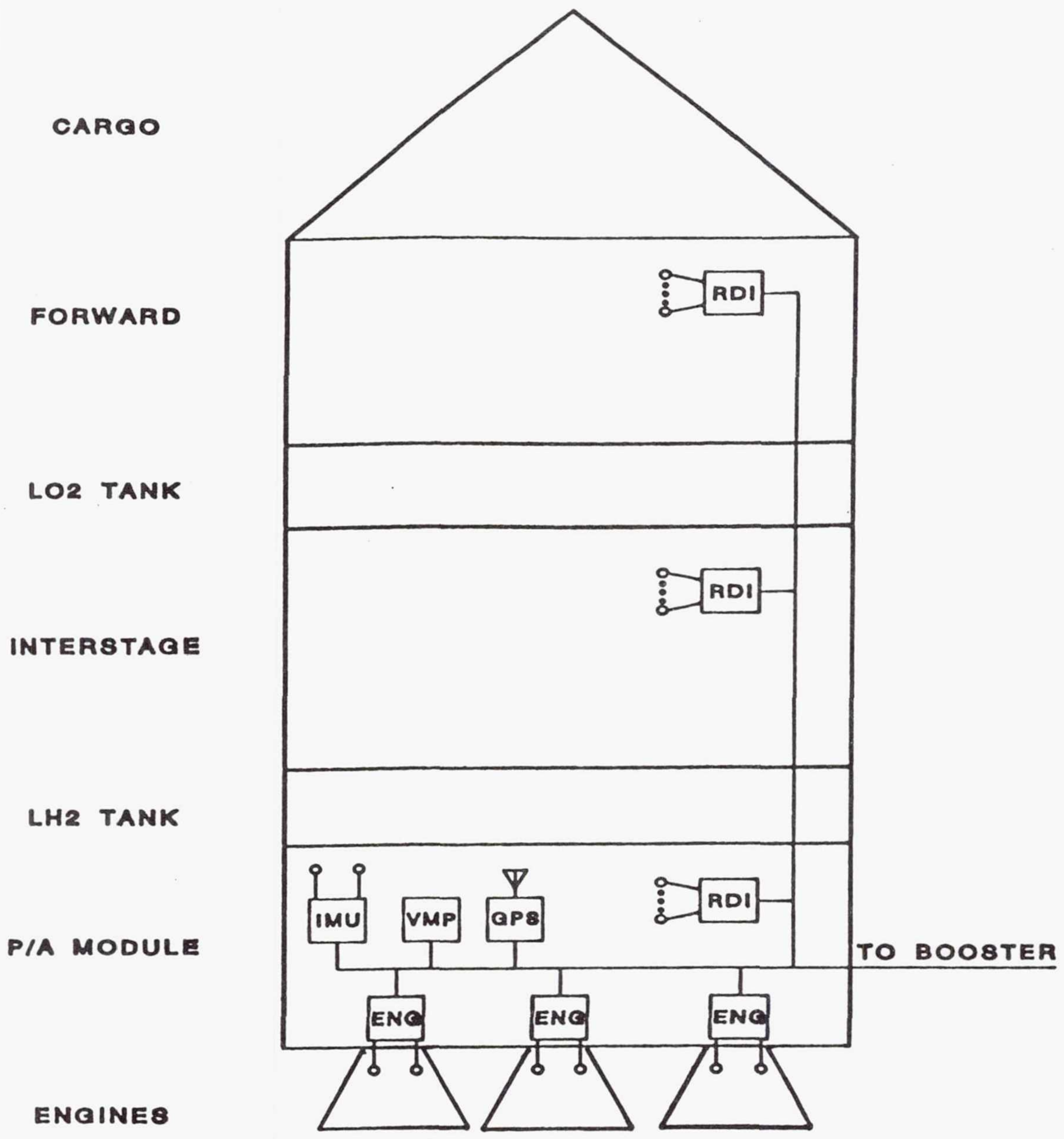


Figure 1.2. General Dynamics Core Avionics Example Configuration

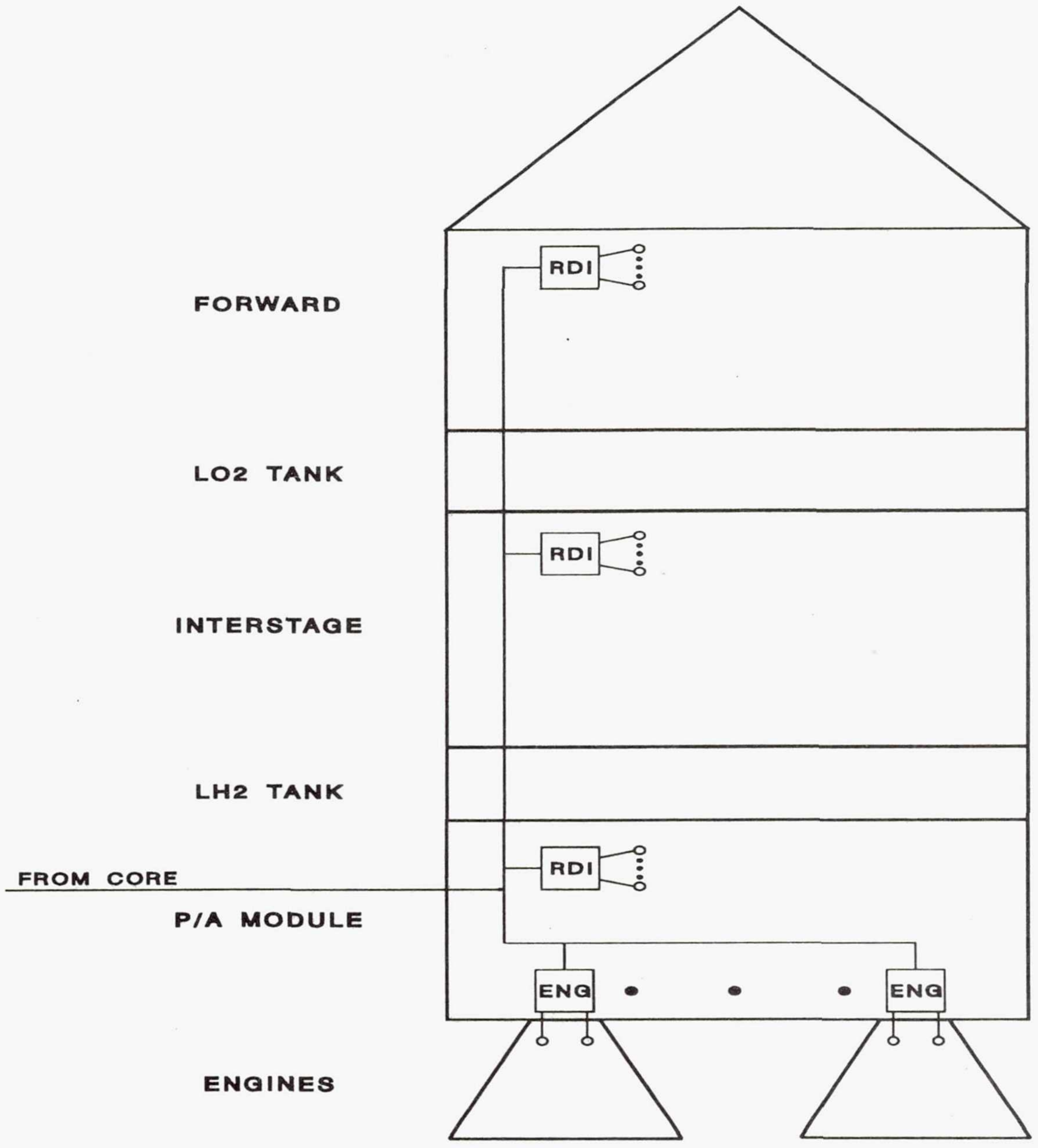


Figure 1.3. General Dynamics Booster Avionics Example Configuration

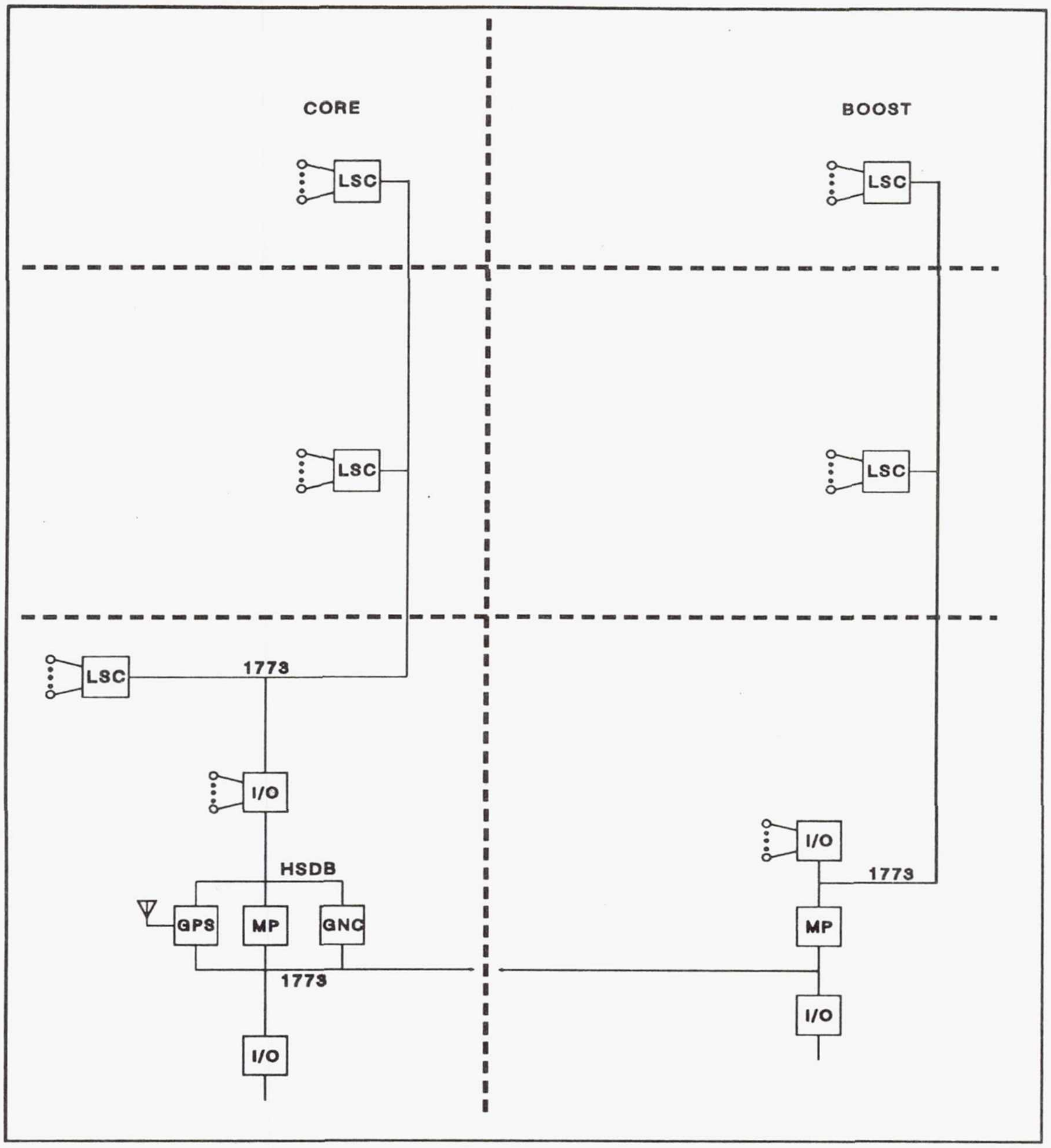


Figure 1.4. Boeing Avionics Example Configuration

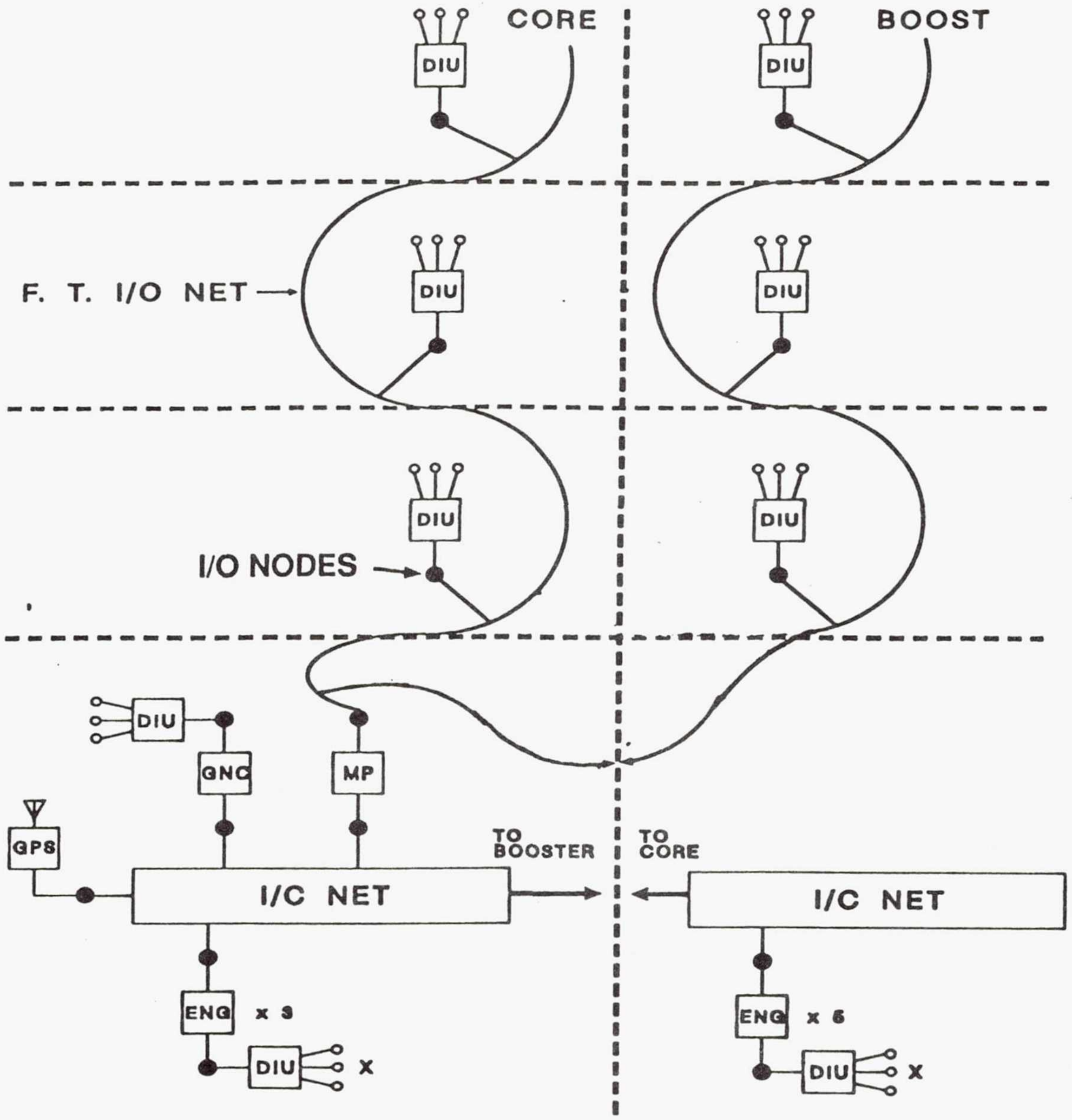


Figure 1.5. AIPS Example Configuration

Module / Function	Enclosures	S.C. Processor/ Memory Module	System Bus I/F Module	Local Data Link Module	Sensor Input Module	Output Module	System Test Module	Power Supply	Sensor I/F Unit Module
Vehicle Management Processor	1	6	3	3			3	3	
Inertial Measurement Unit	1	3	3	3	3		3	3	
Remote Data Interfaces	6	18	18	18	36	36	18	18	
Propulsion	8	24	24	24	48	48	24	24	
TOTAL	16	51	48	48	87	84	48	42	408

Note: Shaded entries represent sensor/actuator I/O.

Table 1.1. Module Counts for General Dynamics Example

Function \ Module	Enclosure	Processor Module	Memory Module	1773 Bus Module	HSDB Module	Fault Tolerance Module	Signal Conditioning Module	Power Supply
GN & C Processor	1	8	2	4	4	4		4
Mission Processor (2)	2	16	4	8	8	8		8
Local Signal Conditioner (42)	42			42			252	84
I/O Channels (16)	16			16	8		80	32
TOTAL	61	24	6	70	20	12	332	128

Note: Shaded entries represent sensor/actuator I/O.

Table 1.2. Module Count for Boeing Example

Module / Function	Enclosure	Power Supply	Processor	Memory	Shared Memory	Net Node	Communicator	Interstage	IOS	ICIS	DIU I/F Control	DIU Signal CNDX	DIU Actuator Module
GN & C (1)	1	3	8	8	4		4	4	8	4			
Mission Management (1)	1	3	8	8	4		4	4	8	4			
Propulsion (8)	8	24	64	64	32		32	32	64	32			
Sensor/Actuator DIU	36	36									36	66	16
IC Network (11)						33							
IO Network (12)						38							
TOTAL	46	66	80	80	40	71	40	40	80	40	36	66	16

Note: Shaded entries represent sensor/actuator I/O.

Table 1.3. Module Count for AIPS Example

the General Dynamic configuration is about 250, in the Boeing configuration about 125 and in the AIPS configuration, assuming only the processor modules use programmable computer, is 80. Given the uncertainty in the configuration and the designs particularly regarding computer requirements for propulsion, these should not be compared. The point of importance to be considered is that a large number of embedded computers are required for this application. All of these computers must interact in such a manner that real-time deadlines are met, maintained and that overall synchronization and timing is maintained. Design and validation methods which account for this complexity must be used to develop MPRAS applications. Furthermore, the Boeing architecture is structured such that all time critical sensors/actuators must reside in the propulsion/avionics (P/A) module.

The Boeing architecture distributes the 2 megabit/sec 1773 bus throughout the vehicle while confining the 50 megabit/sec system busses to each P/A module. The General Dynamics architecture distributes the 50 megabit/sec vehicle management and sensor data network bus throughout the vehicle. Maintaining the acceptable timing and bit error rates for the faster bus over greater physical distances represents a higher technical risk. There are some observations that can be made about specific architectures. If the number of sensors/actuators required are increased, the Boeing architecture limits at 31 the number of remote terminals connected to each of the four transducer busses. Additional sensors/actuators can be connected to the flight control bus but care must be exercised to not violate time criticality constraints associated with the flight control bus.

The I/O net of the AIPS must be distributed throughout the vehicle. The AIPS proof-of-concept I/O network bandwidth is two megabits/sec. This bandwidth is expected to be increased to 20 or 30 megabits/sec. As such, its bandwidth will fall between that of the Boeing and the General Dynamics architecture. While the AIPS I/O network can be configured as linear connections from node to node, it has features which permit use of spare links to reconfigure around failed links and nodes. Use of this capability would represent a more efficient use of the AIPS I/O network hardware. The recommended topology for the AIPS I/O nodes and links is a binary tree with I/O nodes at each branch point and spare links included within the tree. The partitioning of the I/O nodes into the vehicle sections and limitations on the number of links which can cross a boundary between vehicle sections could constrain the I/O network topology to an extent that the recommended topology cannot be fully realizable. The number of sensors and actuators handled by AIPS can be increased by adding I/O nodes and DIU's to the I/O networks. This can be accomplished up to the maximum allowable number of nodes in a network. Beyond this, new networks would have to be set up. The 4000-6000 sensors and actuators required with a maximum combined bandwidth of 1 megabit/sec is well within the capacity of the AIPS.

2. INFORMATION REQUIREMENTS

2.1. Requested Information

General Discussion – The information requested for the assessment and characterization of the MPRAS architectures was baseline MPRAS requirements, hardware module specifications, software operating system specifications, application functional description, descriptions of architectures and topologies to meet the application requirements and descriptions of the fault avoidance and fault tolerance features of each architecture. The information used for this evaluation encompasses all MPRAS parts 1 and 2 information reported through December 1989.

Characterizations capable of quantitatively discriminating between different "well designed" architectures require high-fidelity architectural specifications. Such information is consistent with the later stages of development. In the development stages prior to the Preliminary Design Review (PDR), less precise information is available. Consequently, the characterization of MPRAS architectures of necessity relied upon qualitative evaluation factors, subjective measures and quantitative architectural parameter sensitivity analyses.

The following paragraphs describe the design information requested for this effort.

Specification of Hardware Architecture – The specification of the hardware architecture requested the identification of all hardware functions including:

- power modules
- processor modules
- memory modules
- I/O communication links and modules
- data acquisition and distribution modules
- interprocessor communications links
- data buses
- maintainability features
- test interface modules
- testability features

- fault masking and reconfiguration mechanisms
- error detection mechanisms
- timing/clocking modules

A brief description of each module or element including relevant performance parameters such as processor speed, communications bandwidth, memory read/write times, communications overhead and capacities was requested.

In addition to performance parameters, module failure rates for various mission phases were requested. For design parameters that were not known, the assumed range of values for these parameters was requested.

Specification of the Application and Architectures to Meet Baseline Requirements – The application specification requested called for a functional decomposition. For each subfunction a description of the inputs, processing, outputs and special requirements such as transport delay, jitter and process update rate, was requested. Processing workload and information flow between subfunctions as well as the basis for the workload estimates were requested.

A detailed audit of the sensors and actuators required to support the application was requested. The audit called for specification of each sensor/actuator, the type, proposed redundancy, number of each type, number of bits, source/destination subfunction associated with the sensor/actuator, associated failure rate for each mission phase and on-line test and calibration requirements.

The proposed topology for the hardware elements and interconnections for each vehicle configuration specified in the baseline requirements was also requested. In addition, the allocation of subfunctions to specific hardware resources was requested.

The overall mission reliability and safety requirements as well as the size, weight and power of the MPRAS for each configuration were requested.

Specification of the Reliability and Fault Tolerance Features of the Architecture – Specification of the fault tolerance features of an architecture called for the identification of the set of faults to be tolerated. Fault identification was to include type, frequency of occurrence, dependence on mission phase or technology and a characterization of the errors induced by the fault type. Error effects were to be characterized by:

1. Count: single versus multiple
2. Origin: environmental factors

3. Activity: dormant versus active
4. Duration of activity: transient versus permanent
5. Extent: local versus distributed
6. Temporal behavior for multiples: coincident versus separated
7. Cause of multiples: independent versus common mode

Descriptions of the fault tolerance features such as error detection, error masking, fault containment, consistency, reconfiguration, redundancy, and redundancy management, was requested for each architecture. Recovery strategy descriptions for both the local (subsystem) and global (system) levels were requested. Assumptions for the effectiveness of error detection and for the times required to detect, recover, and reconfigure and the techniques for providing fault-tolerant power, clocking, synchronization and startup were also requested. Descriptions of fault tolerance features called for the identification of those portions of each feature that were to be incorporated in hardware and those portions which were to be provided by software. Also, the approach used to detect faults in the fault tolerance mechanism was requested.

The primary reliability parameters associated with the architecture were requested. These included, but were not limited to, module failure rates, transient frequency and duration, coverage parameters such as latency and detection effectiveness, redundancy of each element and the number of spares.

The techniques expected to be employed for fault tolerance in the application software were to be defined. Methods and policies which were to be employed to avoid software design faults and to provide software quality assurance were to be identified.

Life Cycle Cost Report – A Life Cycle Cost (LCC) report which summarized the recurring LCC analysis for each MPRAS architecture was requested. It was requested the report state all ground rules and assumptions used and identify all factors included in the recurring LCC. Assumptions regarding the use of common modules within the architecture, which are used in other avionics programs or in ALS ground support, were to be stated. Assumptions regarding maintenance and testability were to be included in the report.

System Software Architecture Descriptions – It was recognized that limited information would be available regarding the system software architecture for some of the MPRAS designs. To the extent that this information was available, it would be used. The information of interest regarding system software architecture included a description of control characteristics such as distributed, central or hierarchical and descriptions of functions including task scheduling, I/O services, interrupt services, memory management, utilities, interprocessor communications services and functions related to fault tolerance. In addition, performance information such as function overheads, expected execution time for a given system function, function response times and uncertainty in response times was requested.

MPRAS Engineering Trade Studies and Analyses – The engineering trade studies and analyses conducted by the three MPRAS contractors to determine the requirements and characteristics of the specified architectures were requested.

System Development Methods – The complexity and criticality of the mission requirements dictates the use of rigorous, systematic methods and policies for the development of MPRAS. This is necessary to assure that all system objectives such as cost, performance, reliability, safety and maintainability are met. Good engineering practice dictates that these factors be addressed starting in the concept, requirements and early design phases and continuing throughout the development.

To validate that the design objectives of a complex system are met requires that systematic validation must be conducted throughout the system development starting at the beginning of the development cycle. All elements of the architecture development (application software, operating system software and hardware elements) must be included in the design validation.

The methods expected to be used for the development of MPRAS will determine to a large extent the potential for a successful development. Documentation of the system development methodology that each contractor expected to use was requested.

2.2. Information Sources

Listed below are the information sources used to conduct the architecture characterizations and assessments.

Boeing Aerospace

MPRAS System/Subsystem Requirements Document (SRD),
February 1989, Doc. No. 180-30579-2.
MPRAS Third Quarterly Review, March 1989
MPRAS Conceptual Avionics Architecture Specification,
July 1989, Doc. No. 180-30579-4.
MPRAS Fourth Quarterly Review, July 1989
MPRAS Final Review, October 1989
MPRAS Preferred Vehicle Avionics Architecture Specification,
October 1989, Doc. No. 180-30579-5.

MPRAS Critical Item Development Specification, GN&C Processor, October 1989, Doc. No. 180-30579-7.
MPRAS Critical Item Development Specification, Local Signal Conditioner, October 1989, Doc. No. 180-30579-9.
MPRAS Critical Item Development Specification, I/O Channel, October 1989, Doc. No. 180-30579-10.

General Dynamics Space Systems Division

MPRAS Reference Vehicle/Requirements, June 1989
Preliminary MPRAS Architecture Specification, Vol. 1, Concept Specification, July 1989
Future Launch System Technology, Contracted Exploratory Research & Development, MPRAS Review, July 1989
MPRAS Point Design, Fault Set, August 1989
MPRAS Point Design, Design Evaluation, August 1989
MPRAS Architecture Specification, Vol. I, September 1989
Draft Version, MPRAS Architecture Specification, Vol. II, (Application to Advanced Launch System), September 1989
MPRAS Point Design, Fault Detection, Isolation, and Recovery, September 1989
MPRAS ADP 2103, Technical Interchange Meeting, October 1989
MPRAS Point Design, Performance/Dependability Goals, SCM, July 1989

The Charles Stark Draper Laboratory, Inc.

Advanced Information Processing System: Input/Output Network Management Software, Contract NAS1-17666, May 1988
Completion of AIPS (FY 88 Tasks), Oral Review, March 1989
Validation of Core Fault Tolerance Concepts, AIPS Fault Tolerance Concepts, March 1989
AIPS FTP Reliability Analysis, March 1989
MPRAS 2102: AIPS for ALS Review, Program Overview and Status October 1989

Other Information

Architecture Specification for PAVE PILLAR Avionics,
SPA90099001A, January 1987

Modular Avionics, IBM Product Specifications for JIAWG Modules

Application of Fault Tolerance Technology, Vol. I, Design of
of Fault-Tolerant Systems, SDIO BM/C^3 Processor and Algorithm
Working Group, Avizienis, A. and Gilley, G., Editors

Application of Fault Tolerance Technology, Vol. II, Management
Issues: Contractor Milestones and Evaluation, SDIO BC/C^3
Processor and Algorithm Working Group, Rennels, D.
Gilley, G., Editors

RADC Testability Notebook, RADC-TR-82-189, Hughes Aircraft
Reliability Handbook, MIL-HDBK-217E

3. RELIABILITY, AVAILABILITY, MAINTAINABILITY AND TESTABILITY EVALUATION

3.1. Introduction

Reliability and the closely related characteristics of maintainability, availability, testability and fault tolerance are of prime importance in the design of the MPRAS architecture. Not only does improved reliability lead to cost reductions through reduced logistics support costs, it enables introduction of functionality into the avionics that will support reduced integration testing and operations costs.

The avionics used in current launch vehicles provide the basic mission functions with moderate reliability. Improvements in the reliability of the basic systems would result in reduced life cycle costs through reduced support costs. The avionics necessary to provide functionality such as adaptive, guidance, navigation and control (AGN&C) integrated health monitoring, and increased vehicle autonomy for a complex vehicle could require thousands of sensors and in excess of 100 electronics modules. If the average module MTBF for the prelaunch environment was 100,000 hours, the reliability of the avionics at the end of 200 hours on the launch pad would be about 0.8. If the average sensor MTBF was 50,000 hours, it would be reasonable to expect 10 or so sensor failures at the end of 200 hours on the pad. Fault-tolerant avionics architectures are required to provide the advanced functional capabilities while at the same time meeting reliability requirements.

Consider using a simple fault-tolerant architecture with three independent redundant channels whose output is voted. If the overall unreliability must be less than 1 x 10^{-5} and the exhaustion of components is the primary failure mechanism, the probability of failure for each channel must be less than about 1.82 x 10^{-3}. If 100-200 modules are required in each channel, the average module failure rate must be between 5 x 10^{-5}/hour and 10^{-4}/hour for 10 minutes in the missile launch environment.

Even if it is assumed that this failure rate is the MIL-HDBK-217E predicted failure rate and that it is not necessary to account for the transient failures, a higher rate than the 217E rate, the failure rate/module for ground-fixed environment must be between 10^{-5} and 2x10^{-5}/hour.

This failure rate is at or below that for modules of the complexity of processors, memory, and I/O controllers implemented using current VLSI technologies. One must conclude that it would be difficult to achieve the desired reliability with a simple independent redundant 3 channel system.

Figure 3.1 diagrams a framework for reliability evaluation. Evaluation starts at the mission requirements level and incorporates the system architecture, the technology of implementation, and the appropriate failure processes.

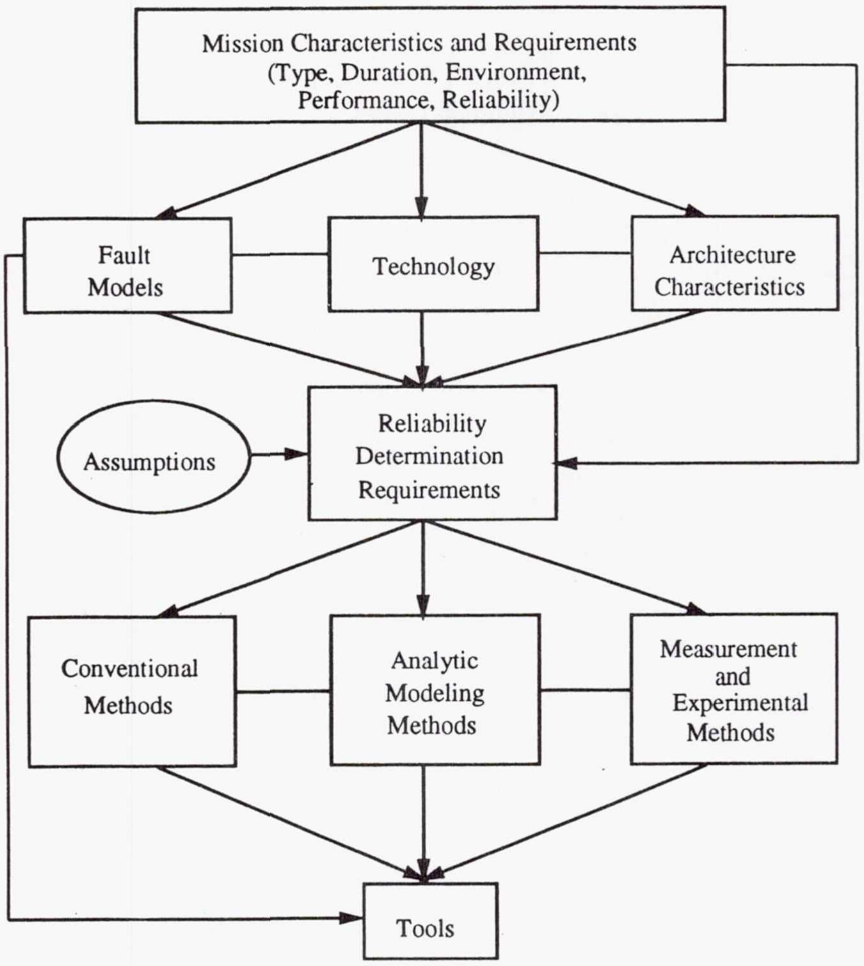

Mission Characteristics and Requirements (Type, Duration, Environment, Performance, Reliability)
Fault Models
Technology
Architecture Characteristics
Assumptions
Reliability Determination Requirements
Conventional Methods
Analytic Modeling Methods
Measurement and Experimental Methods
Tools

Figure 3.1. Framework for Reliability Evaluation

The three areas of focus for reliability determination are thus: 1) missions, fault models, architectures and technologies; 2) reliability determination requirements; and 3) methods and tools.

The MPRAS application characteristics which impact reliability modeling requirements are mission criticality, many distinct mission phases with diverse activity levels and reliability requirements, a relatively large number of distributed system modules and a harsh operating environment.

The high system reliability requirements dictate the need for a fault-tolerant system. Such systems have to be carefully evaluated to determine whether or not the requirements are met. Fault tolerance mechanisms makes evaluation more difficult by increasing the number and complexity of significant factors affecting system reliability.

Except for the 10 minute launch phase, mission profiles for all three architectures differ significantly. Table 3.1 summarizes the various phases and their duration.

One of the most important differences is the amount of time that the avionics is powered prior to launch. There is a factor of 8 between the shortest and the longest time.

BOEING			GENERAL DYNAMICS			MARTIN/DRAPER		
Phase	Hrs On	Hrs Off	Phase	Hrs On	Hrs Off	Phase	Hrs On	Hrs Off
Recovery	8	2	Fctry Chckt	136	~800	Rec&Refrbsh	200	
Refurbish	150	150						
Tnk,P/A Mte	8	16	Vert Integ	10	~100	Integ&Rllt	400	
Cr,Bst Mate	32	88						
Payld Integ		90	Cargo Integ	6	~100			
Launch Pad	48	32	PreLaunch	25	~100	PreLaunch	200	
Lnch/Ascnt	0.15		Flight	.17		Ascent	.17	
			Core Pow Flt					
			C Eng Shtd					
Orbit	12.65		On-Orbit			On-Orbit	1.5	
			Payld Sprtn					
			Core DeOr					
RecModRbst	0.1		PA Mod Sprtn			Flyback	.17	
RecModFlt	33.2		PA Recovery					
			BRM Sprtn					
			BRM Recvry					

Table 3.1. Mission Phases

This difference would result in significantly different failure states at launch time and directly impacts the availability of the avionics for launch.

Various reliability requirements have been put forth for MPRAS applications. Boeing indicates a requirement of 0.999998 for the first flight of the core recoverable P/A module and a 0.9999 for the 50th and last flight for that module. A reliability of 0.9999997 is allocated to computing and 0.9999984 is allocated to I/O. General

Dynamics has put forth a requirement of 0.99999 at the end of a 10 minute launch phase with the assertion that reliabilities better than 0.9999 are not likely to be cost effective. General Dynamics has set a pre-launch availability goal of 0.99954 and 100% fault detection for all test methods combined with 99% from built-in-test features of the VLSI chips.

The fault models for the reliability analysis will include transient and permanent faults. The failure rates for the modules will be for the most part based on MIL-HDBK-217E. Permanent failures will use the base rates adjusted by the appropriate microelectronics environmental factor for the mission phase of interest. Allowance for transient failures occurring at rates faster than the permanent failure rate will be made.

Other fault models that should eventually be taken into account for MPRAS reliability evaluations include the common mode failure. The common mode failure violates the independent failure assumption upon which the reliability of redundant systems is based. Mechanisms need to be included in the design to cope with important subsets of common mode failure mechanisms. Since there may be up to 200,000 connector contacts in an MPRAS application, other fault mechanisms such as intermittent failures due to mechanical vibration should also be included. The electromagnetic environment (EME) within which the MPRAS system operates can pose a significant threat to the safe operation of these systems. Whether by radiation or conduction, electrical energy from on-board components, radar equipment, and RF communications equipment and from external sources such as lightning strikes, high energy radio frequency broadcasts (HERF), nuclear radiation and electrostatic discharge can couple into digital microelectronics and induce faults. Digital microelectronic systems are particularly vulnerable to this coupled energy because of fast circuit switching times, the relatively small amount of energy required to upset their operation and the fact that critical operating state information stored in registers and memories can be easily lost.

The NASA Langley reliability modeling support tools ASSIST and SURE were used for reliability analyses.

3.2. Hardware Module Failure Rates

Projected failure rates for the General Dynamics and Boeing MPRAS hardware modules were not reported in the MPRAS documentation. Boeing, however, specified reliability objectives for each major hardware unit. Since JIAWG or JIAWG-like modules were proposed for most of the hardware functions, it was decided to use the failure rates for these modules as the basis for reliability analyses.

The MIL-HDBK-217E-predicted failure rates that were available for JIAWG modules of interest are shown in Table 3.2. These failure rates have been translated to the Ground Fixed environment. The Ground Fixed environment will be used for the pre-launch on-pad portion of the MPRAS mission. The assumed failure rates for General Dynamics hardware modules are given in Table 3.3. The rate for the self-checking processor is an estimate which assumes that two processors, two 1 megaword memories and two program ROM's are on the board along with a single test and maintenance processor and a self-checking comparator. This estimate assumes a memory failure rate of 1.6 x 15^{-5}/hr for 1 megaword memory when added to an existing module. General Dynamics did not address the feasibility of this degree of complexity for a module or if a 10 MIP self-checking processor can be designed to have a power consumption of 10 to 15 watts. The General Dynamics Local Data Link has not been defined. Its failure rate will be assumed to be the same as a JIAWG data processing module.

Table 3.4 gives the failure rates projected for the Boeing hardware modules. The Fault Tolerance Module for the Boeing architecture was not well defined. Based on rough descriptions of its functionality and a block diagram, it is judged to be of the same complexity as the Processor. It has less memory but has the added function of a voter, cross-channel interfaces and a bus switching unit.

The AIPS FTP failure rates have been reported for a VLSI implementation. The failure rates include an I/O processor and a computation processor along with two floating point co-processors, a megabyte of PROM and RAM memory and associated glue logic. Table 3.5 details these failure rates for the Ground Fixed environment factor.

In order to get the FTP failure rate more or less on an equivalent footing with the failure rates used for the processors used in the Boeing and General Dynamics architectures, the FTP memory failure rates were modified to be consistent with those used in the JIAWG modules. These are:

- 1.6 x 10^{-5}/hr for adding 1 megaword RAM to an existing module
- 5 x 10^{-6}/hr for adding $\frac{1}{2}$ megaword PROM to an existing module

An additional 6 x 10^{-6}/hr was added to cover the addition of a test and maintenance interface.

The failure rate for the remaining FTP functions such as the Input/Output Sequencer, the Intercomputer Communications Interface Sequencer, the shared memory, the Communicator (voter) and Dual Port Memory are not known. However, assumptions were made for them based on rough estimates of function

Module	Failure Rate (Ground Fixed)
Processor 3.8 MIPs 1750A processor Power: 21 watts 512K word RAM 16K Word EEPROM Test/Maintenance Processor	2.4×10^{-5}/hr
Power Supply Module 5v @ 44 amps	8×10^{-6}/hr
1553B Interface 1750A processor Power: 21.5 watts 128K word RAM 16K Word EEPROM Test/Maintenance Processor	1.8×10^{-5}/hr
High Speed Bus 1750A processor Power: 31.5 watts 128K word RAM 16K Word EEPROM Test/Maintenance Processor	2.4×10^{-5}/hr
Bulk Memory 512K word EEPROM Power: 15 watts 16K Word EEPROM Test/Maintenance Processor	1.4×10^{-5}/hr

Table 3.2. MIL-HDBK-217E Failure Rates for JIAWG Modules

Module	Failure Rate (Ground Fixed)
Self-checking Processor	6×10^{-5}/hr
Local Data Link (Voter)	2.4×10^{-5}/hr
High Speed Data Bus	2×10^{-5}/hr
Power Supply	8×10^{-6}/hr

Table 3.3. Failure Rates for General Dynamics Modules

Module	Failure Rate
Processor	2.4×10^{-5}/hr
Memory	2.8×10^{-5}/hr
Fault Tolerance Module (Voter)	2.4×10^{-5}/hr
Power Supply	8×10^{-6}/hr
High Speed Data Bus I/F	2×10^{-5}/hr
1773 I/F	1.8×10^{-5}/hr

Table 3.4. Failure Rates for Boeing Modules

Component	Failure Rate (Ground Fixed)
Processors (IOP, CP & 2 floating points (and glue)	11 x 10^{-6}/hr
RAM (BiPolar)(1Megabyte)	320 x 10^{-6}/hr
PROM (Megabyte)	35 x 10^{-6}/hr

Table 3.5. FTP Failure Rates

complexity. That is, there is no basis other than engineering judgement for their values. Table 3.6 gives the failure rates used for the AIPS modules.

Figure 3.2 gives the projected failure rates for processing channels in each of the architectures based on the rates for component modules. A key parameter in determining the reliability of MPRAS architectures during the launch phase is the transient failure rate during this phase. For this evaluation, this rate was assumed to be in the range of 2 to 10 times the 217E rate for the missile launch environment.

The sensor interface hardware was the least defined component for all of the proposed architectures. Consequently, estimates of the failure rates were made. It was assumed that all implementations would involve the use of similar components. It was further assumed that the General Dynamics Sensor Interface module required fewer parts since it was part of a larger assembly (RDI). Consequently, it shared certain common components such as power supplies with other functions, whereas the Boeing Local Signal Conditioner and presumably the Draper Device Interface Unit are self contained. The items included in the estimate are A/D converters, Bus I/F's, Multiplexers, 30 Differential Amplifiers, Power Supplies (low current) and test electronics. Table 3.7 shows the failure rates used for the sensor interface.

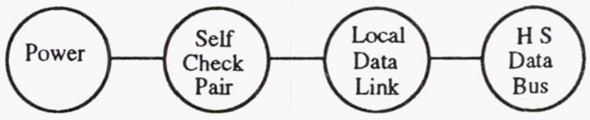
The failure rate estimates are based on a minimum of design information and were made by RTI for the purpose of doing rough overall system reliability comparisons. These estimates are believed to be reasonable but are not represented as accurate. It is assumed that the three contractors can implement the function with equivalent failure rates. The sharing of common components across several functions inherent in the General Dynamics RDI modularity is reflected by the lower failure rate for the

Component	Failure Rate (Ground Fixed)
Processors (IOP, CP, Memory)	3×10^{-5}/hr
Shared Resources (Data Xchg, Memory)	2.5×10^{-5}/hr
I/O Sequencer	2×10^{-5}/hr
IC I/F Sequencer	2×10^{-5}/hr
Power Supply	8×10^{-6}/hr
I/O Node	2×10^{-5}/hr

Table 3.6. AIPS Module Failure Rates

GENERAL DYNAMICS

(1 megaword RAM)



Total /10^6 hrs (GF)

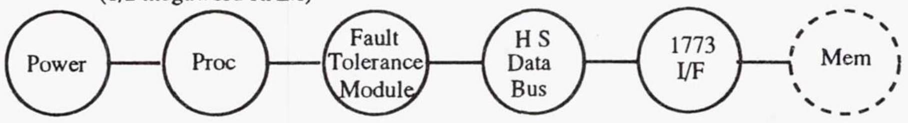
λ = 8 60 24 20

112

Note: Not clear how to add memory w/o adding SCP's

BOEING AEROSPACE

(1/2 megaword RAM)

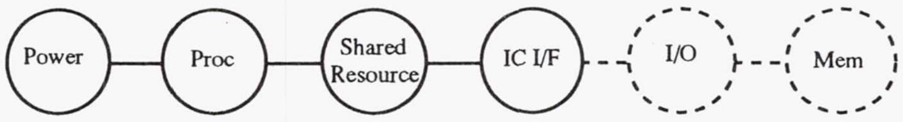


94 + Mem

λ = 8 24 24 20 18 28/

AIPS

(1 megabyte)



83 + I/O
+ Mem

λ = 8 30 25 20 20 28/

Note: Assumes that memory modules can be added

Figure 3.2. Core Processing Single Channel Failure Rates

Component	Failure Rate (Ground Fixed)
Local Signal Conditioner (Boeing 30 inputs)	2.6×10^{-5}/hr*
Device Interface Unit (AIPS 30 inputs)	2.6×10^{-5}/hr*
Sensor Interface Module (General Dynamics 30 inputs)	1.5×10^{-5}/hr*

*Note: This rate is 30% higher than the design objective of 0.999 reliability for a 48 hour mission called out for the Boeing Local Signal Conditioner.

Table 3.7. Sensor Interface Failure Rates

Sensor Interface Module. Since these modules will be used extensively in MPRAS, they can influence overall system reliability. Consequently, good reliability estimates for these modules should be a priority for the MPRAS module design specifications.

3.3. Core Processing Functions

Preliminary Discussion – Core processing resources for the MPRAS applications will be made up of a number of distributed processing centers. Depending upon which architecture and which design options are exercised, these processing centers may be configured as separate multi-channel redundant processors or configured such that all processing centers function as a multi-channel redundant system.

As discussed in the previous section, the MPRAS mission is multi-phased. The reliability of the on-pad and launch phases of the mission is the focus of the reliability analyses discussed in this section. Analysis was carried out for a quadruplex and triplex for both the on-pad and launch phases. Reliability models for these cases were built using ASSIST and analyzed using SURE.

Prior to describing these models in detail, an approximate analysis of the essential parts of these models will be reviewed.

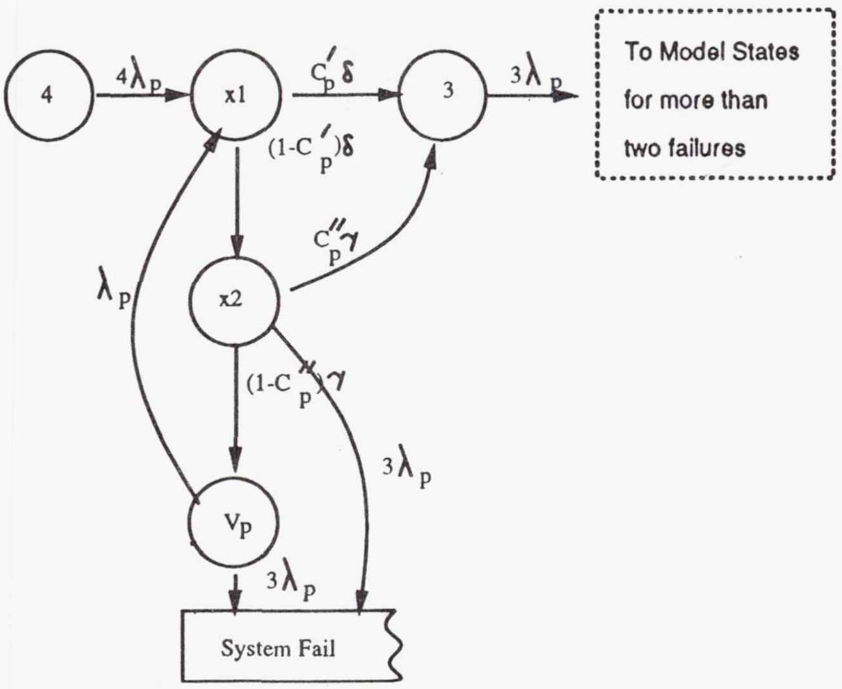


Figure 3.3. Quadruplex On-pad Model

Consider a quadruplex prior to launch. In state 4 of Figure 3.3, all redundant channels are functioning properly. Upon failure, and assuming that no near-coincident second failures occur, a relatively fast recovery process successfully isolates the faulty channel with a probability of C'_p. For those faults that are not detected and isolated by the fast process, a slower, more thorough fault detection process (pre-launch diagnostics) may detect and isolate the faulty processor. Otherwise, the system enters a state (V_p) where a fault has occurred but has not been detected. In this state, the system is believed to be fully functional. The system leaves this state if another fault occurs in the faulty processor, or if a fault occurs in another processor. In state V_p, the system is vulnerable to additional failures. If the vehicle is launched while in this state, the next fault that occurs could result in loss of the vehicle. While a latent fault that could not be detected and isolated with pre-launch diagnostics may not be error-producing after another fault occurs, the conservative assumption is that it will be and as such leads to system failure. Assuming that δ and γ are much faster than the failure rate λ, the probability of being in state V_p is approximated by:

$$P_Q[V_p] \simeq 4\lambda_p(1 - C_p)t_p \qquad (3.1)$$

where: λ_p = pad failure rate
C_p = effective coverage of fast and slow fault detection
t_p = time on pad

For a triplex, equation 3.1 becomes:

$$P_T[V_p] \simeq 3\lambda_p(1 - C_p)t_p \tag{3.2}$$

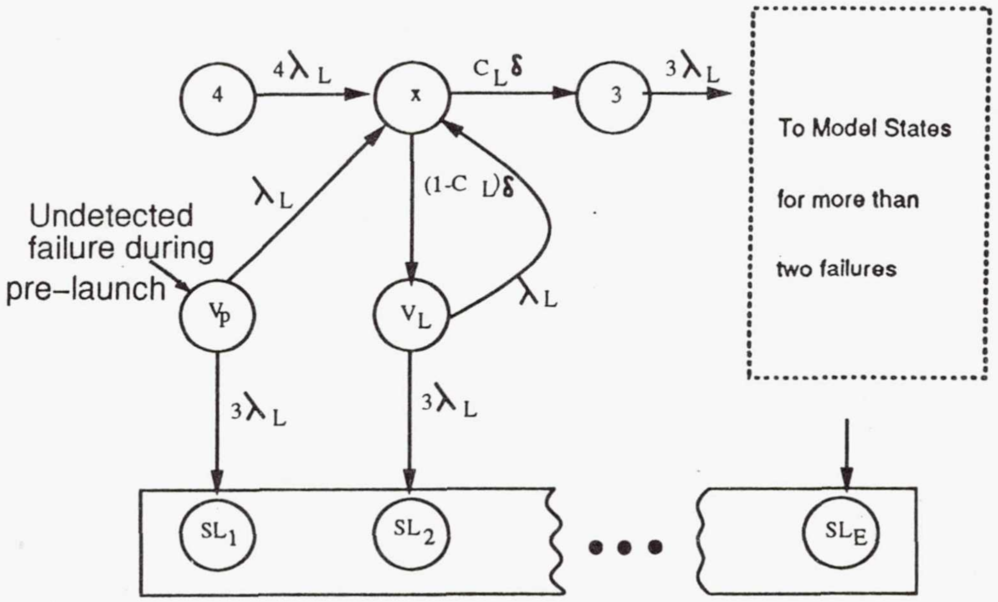


Figure 3.4. Quadruplex Launch

Figure 3.4 shows a portion of a reliability model for the launch. The states are defined as follows:

State	Description
4	All 4 channels operational at beginning of launch interval
Vp	An undetected channel failure occurred during the on-pad phase
x	Recovery state
V_L	Undetected channel failure during launch interval
3	3 operational channels with a failed channel properly eliminated. System can be in this state at beginning of launch due to an on-pad failure or due to a failure after launch
SL_x	System failure states

During launch only a relatively fast fault detection and isolation process is assumed. As long as the fault detection and isolation is not successful, the system is vulnerable to additional faults. If the system starts in a vulnerable state from the pre-launch phase, it remains vulnerable to additional faults. For this analysis, it is assumed that after launch any fault which occurs when the system is in a vulnerable state will result in the loss of the vehicle. Vulnerable states associated with transitions out of the three-operational channel state are included in the model but will not be analyzed for this discussion.

The probability of reaching state SL_1, that is, the probability of losing the vehicle due to incurring additional faults after having launched in a vulnerable state, is approximated by:

$$P_Q[SL_1] \simeq P_Q[V_p]3\lambda_p \left(\frac{\pi_{el}}{\pi_{ep}} \cdot (1 + F_t)\right) t_L \tag{3.3}$$

where:	π_{el}	=	the failure rate environment factor for missile launch
	π_{ep}	=	the failure rate environment factor for ground fixed form from MIL-HDBK-217E
	F_t	=	the ratio of the rate of occurrence of transient faults to the permanent fault failure rate
	t_L	=	duration of the launch phase.

Substituting equation 3.1 for $P_Q[V_p]$ yields:

$$P_Q[SL_1] \simeq 12\lambda_p^2 t_p t_L[1 - C_p]\frac{\pi_{el}}{\pi_{ep}}(1 + F_t) \tag{3.4}$$

Similarly, the probability of vehicle loss for a triplex due to launching in a vulnerable state is given by:

$$P_t[SL_1] \simeq 6\lambda_p^2 t_p t_L[1 - C_p]\frac{\pi_{el}}{\pi_{ep}}(1 + F_t) \tag{3.5}$$

The probability of reaching state SL_2 is approximated by:

$$P_Q[SL_2] \simeq 6\lambda_p^2 t_L^2 \left[\frac{\pi_{el}}{\pi_{ep}}(1 + F_t)\right]^2 (1 - C_L)P[state\ 4\ at\ launch] \tag{3.6}$$

Substituting for $P[state\ 4\ at\ launch]$,

$$P_Q[SL_2] \simeq 6\lambda_p^2 t_L^2 \left[\frac{\pi_{el}}{\pi_{ep}}(1 + F_t)\right]^2 (1 - C_L)e^{-4\lambda_p t_p} \tag{3.7}$$

Similarly, for a triplex

$$P_T[SL_2] \simeq 3\lambda_p^2 t_L^2 \left[\frac{\pi_{el}}{\pi_{ep}}(1 + F_t)\right]^2 (1 - C_L)e^{-3\lambda_p t_p}. \tag{3.8}$$

The probability of vehicle loss for a quadruplex due to exhaustion of components, assuming that a duplex cannot be recovered and that the system starts in state 4, is approximated by:

$$P_Q[SL_E] \simeq 12\lambda_p^3 t_L^3 \left(\frac{\pi_{el}}{\pi_{ep}}(1 + F_t)\right)^3. \tag{3.9}$$

Similarly, the exhaustion probability for a triplex becomes:

$$P_T[SL_E] \simeq 3\lambda_p^2 t_L^2 \left(\frac{\pi_{el}}{\pi_{ep}}(1 + F_t)\right)^2. \tag{3.10}$$

Comparison of the relative magnitude of each of these quantities is of interest. Consider the ratio of $P_Q[SL_1]\ :\ P_Q[SL_2]$:

$$\frac{P_Q[SL_1]}{P_Q[SL_2]} = 2\frac{t_p}{t_L}.\frac{[1 - C_p]}{[1 - C_L]}.\frac{\pi_{ep}}{\pi_{el}}.\frac{1}{(1 + F_t)}.\frac{1}{e^{-4\lambda_p t_p}} \tag{3.11}$$

This ratio ranges from about 0.5 to about 100. For the following parameter ranges of interest:

$$\begin{aligned}
25 \text{ hrs} \le \ & t_p \le 200 \text{ hrs}, \\
& t_L = 0.167 \text{ hrs} \\
0.9 \le \ & C_p \le 0.99, \\
0.9 \le \ & C_L \le 0.99, \\
& C_p \ge C_L, \\
& \tfrac{\pi_{ep}}{\pi_{el}} = \tfrac{2.5}{13}, \\
& F_t = 10, \\
& 10^{-4} \le \lambda_p \le 10^{-3}
\end{aligned}$$

Note that $C_P = .99$ is consistent with the built-in-test design goal for fault detection of the General Dynamics MPRAS modules. For short pre-launch phases combined with lower but reasonable coverage of failures during launch, the two modes of failure contribute more or less equally to system failure. As the pre-launch phase is lengthened, the failures due to launching in a vulnerable state dominate. This dominance can be reduced by improving the coverage of the pre-launch tests.

Consider the ratio of $P_Q[SL_1]$: $P_Q[SL_E]$:

$$\frac{P_Q[SL_1]}{P_Q[SL_E]} = \frac{[1 - C_p]t_p}{\lambda_p t_L^2} \left[\frac{\pi_{ep}}{\pi_{el}} \frac{1}{(1+F_t)} \right]^2 \tag{3.12}$$

For the parameter ranges of interest, this ratio varies from 2.8 to 2200. That is, the probability of failure during launch due to lack of coverage in the pre-launch testing dominates the probability of launch failure due to exhaustion of components.

The second exhaustion of components failure mode for the quadruplex during launch occurs when the system is reconfigured to triplex prior to launch. The probability of failure for a degraded quadruplex due to exhaustion of components is approximated by:

$$P_{QD}[SL_E] \simeq 3\lambda_p^2 t_L^2 \left(\frac{\pi_{el}}{\pi_{ep}} (1+F_t) \right)^2 P_Q[\textit{degenerates to 3 on pad}] \tag{3.13}$$

Substituting an approximation for the probability of degrading to triplex during pre-launch, equation 3.13 becomes:

$$P_{QD}[SL_E] \simeq 3\lambda_p^2 t_L^2 \left(\frac{\pi_{el}}{\pi_{ep}} (1+F_t) \right)^2 (1 - e^{-4\lambda_p t_p}) C_p \tag{3.14}$$

Consider the ratio of $P_Q[SL_1]$: $P_{QD}[SL_E]$;

$$\frac{P_Q[SL_1]}{P_{QD}[SL_E]} = 4 \left(\frac{t_p}{t_L} \right) \frac{\pi_{ep}(1 - C_p)}{\pi_{el}(1+F_t)C_p(1 - e^{-4\lambda_p t_p})} \tag{3.15}$$

This ratio ranges between 1 and 117 for the parameter ranges of interest. The loss of an avionics processing site due to lack of coverage in the pre-flight diagnostics is about the same as or is much greater than the exhaustion of components.

Consider the ratio $P_{QD}[SL_E : P_Q[SL_E]$:

$$\frac{P_{QD}[SL_E]}{P_Q[SL_E]} = \frac{C_p}{4\lambda_p t_L} \cdot \frac{\pi_{ep}}{\pi_{el}} \cdot \frac{1}{(1+F_t)} \cdot (1 - e^{-4\lambda_p t_p}) \tag{3.16}$$

This ratio ranges between about 2 and 20 for parameter ranges of interest.

From these simple calculations, it can be determined that for the ranges of parameters assumed, the two main contributors to the failure of an avionics processing site are: 1) failure due to launching with undetected failures and 2) failure due to attrition of components from starting launch with a quadruplex degraded to a triplex.

Self-Checking Pair with a Spare – The self-checking pair (SCP) with a spare is the basic processing kernel for each channel of the General Dynamics architecture. An SCP has a failure rate that is nominally twice as fast as that for a single processor. However, the SCP should result in higher fault coverage and reduced fault latency than methods that rely only on background diagnostics, memory tests and voting of computed application data with that from redundant computing channels.

A Markov model for a self-checking pair architecture with one spare is illustrated in Figure 3.5. This model contains states and transitions to capture the occurrence of both permanent and transient faults in the active pair and in the spare pair. It also contains states and transitions to model the detection of a fault in the SCP and the replacement of the faulty SCP by the spare SCP and to model the return of a processor to the spare pool after experiencing a transient fault.

The states in this model can be represented by a vector

S=(SCP,S,T,P), where

SCP	= the number of operational SCPs
S	= the number of operational spare SCPs
T	= 1 if the fault is transient, 0 otherwise
P	= 1 if the fault is transient, 0 otherwise

Transitions from one state to another in the model are made according to the following rates and probabilities:

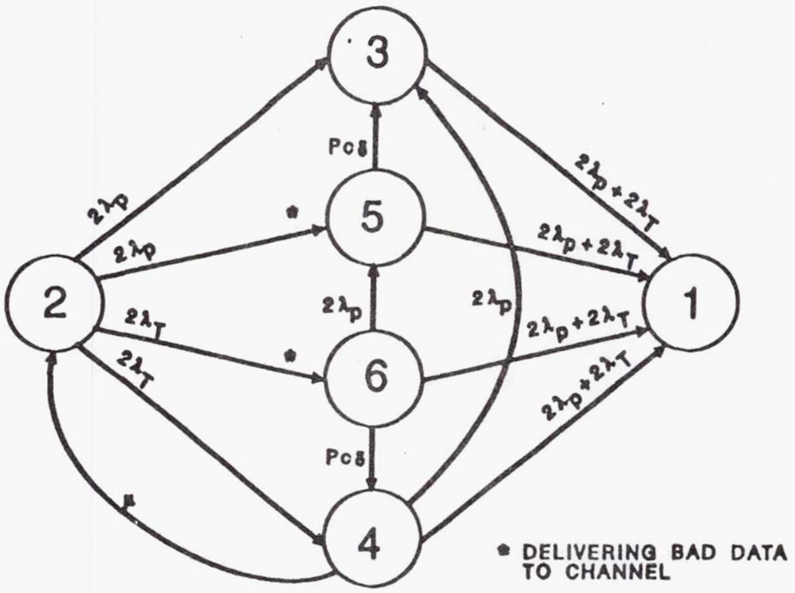

3
5
2
6
1
4
$2\lambda_P$
$2\lambda_P$
$2\lambda_T$
$2\lambda_T$
$P_c\delta$
$P_c\delta$
$2\lambda_P$
$2\lambda_P$
*
*
μ
$2\lambda_P + 2\lambda_T$
$2\lambda_P + 2\lambda_T$
$2\lambda_P + 2\lambda_T$
$2\lambda_P + 2\lambda_T$
* DELIVERING BAD DATA TO CHANNEL

Figure 3.5. Self-Checking Pair with Spare Model

λ_P	Permanent Processor Failure Rate
λ_T	Transient Processor Failure Rate
P_c	Probability that fault is detected and spare is successfully switched in to replace failed SCP
δ	Rate at which fault is detected and spare is switched in to replace failed SCP
μ	Rate at which SCP failed due to a transient fault is recovered

The states in this launch model are described in the following table:

State Number	State Vector	Description
2	(1,1,0,0)	1 operational SCP 1 available spare SCP
3	(1,0,0,1) Spare unavailable due to permanent fault	1 operational SCP
4	(1,0,1,0) Spare unavailable due to transient fault	1 operational SCP
5	(0,1,0,1)	0 operational SCPs due to permanent fault 1 available spare SCP
6	(0,1,1,0)	0 operational SCPs due to transient fault 1 available spare SCP
1	(0,0,0,0)	0 operational SCPs 0 operational spare SCPs

This model was analyzed for both the on-pad and launch mission phases.

Figure 3.6 shows the probability of failure for a self-checking pair with a spare as a function of mission time and as the coverage factor, P_c, is varied. The transient failure rate is assumed to be zero and the permanent failure rate, λ_p, is assumed to be 6 x 10^{-5}/hour. When transients are included at a rate of $10\lambda_p$, the unreliability results are increased by about a factor of 10. This is due to the modeling assumption that transient faults that occur after either the primary or spare SCP has failed cannot be recovered and results in the channel failure. This assumption is appropriate for launch. Prior to launch this SCP could be recovered.

For a permanent failure rate of 3 x 10^{-5}/hr for each processor in the SCP, the average failure rate for an SCP with a spare is 1.32 x 10^{-6}/hr for a 200 hour mission and is about 7 x 10^{-7}/hr for a 25 hour mission. These compare to 6 x 10^{-5}/hr for an unspared SCP. Note that this improvement in reliability is accomplished through the use of four single processor units with memory.

Quadruplex On-pad – A Markov model for a quadruplex architecture during the on-pad mission phase is illustrated in Figure 3.7. This model includes fault occurrence and fault handling states for both permanent and transient faults. For both permanent and transient faults, the model contains fault handling states for both fast detection/isolation/and recovery mechanisms such as switching out out-voted processors and slower mechanisms such as memory scrubs. For example, state 11 represents a fast FDIR process after the occurrence of a permanent fault. This process leads to a recovered state where the failed processor is switched out (state 13) or to a slower diagnostic process (state 14). The slower diagnostic process state can lead to system failure if another fault occurs, to the recovered state (state 13), or to an undetected fault state (state 19).

Since this is a model of system behavior during the on-pad mission phase, system failure has to include the inability to launch. States 10 - 23 in Figure 3.7 are operational states, and states 1 - 9 represent system failure states and those operational states that preclude vehicle launch. Of the states from which a launch is possible, the following are of sufficiently high probability to be included in the launch-phase model: state 10 (4 nonfaulty processors), 13 (3 nonfaulty processors), 19 (4 operating processors, one with an undetected fault), and 23 (3 operating processors, one with an undetected fault).

The states in this model can be represented by a vector

S=(N,P,T,R,F,UPF,UTF), where

N = the number of operational processors in the state

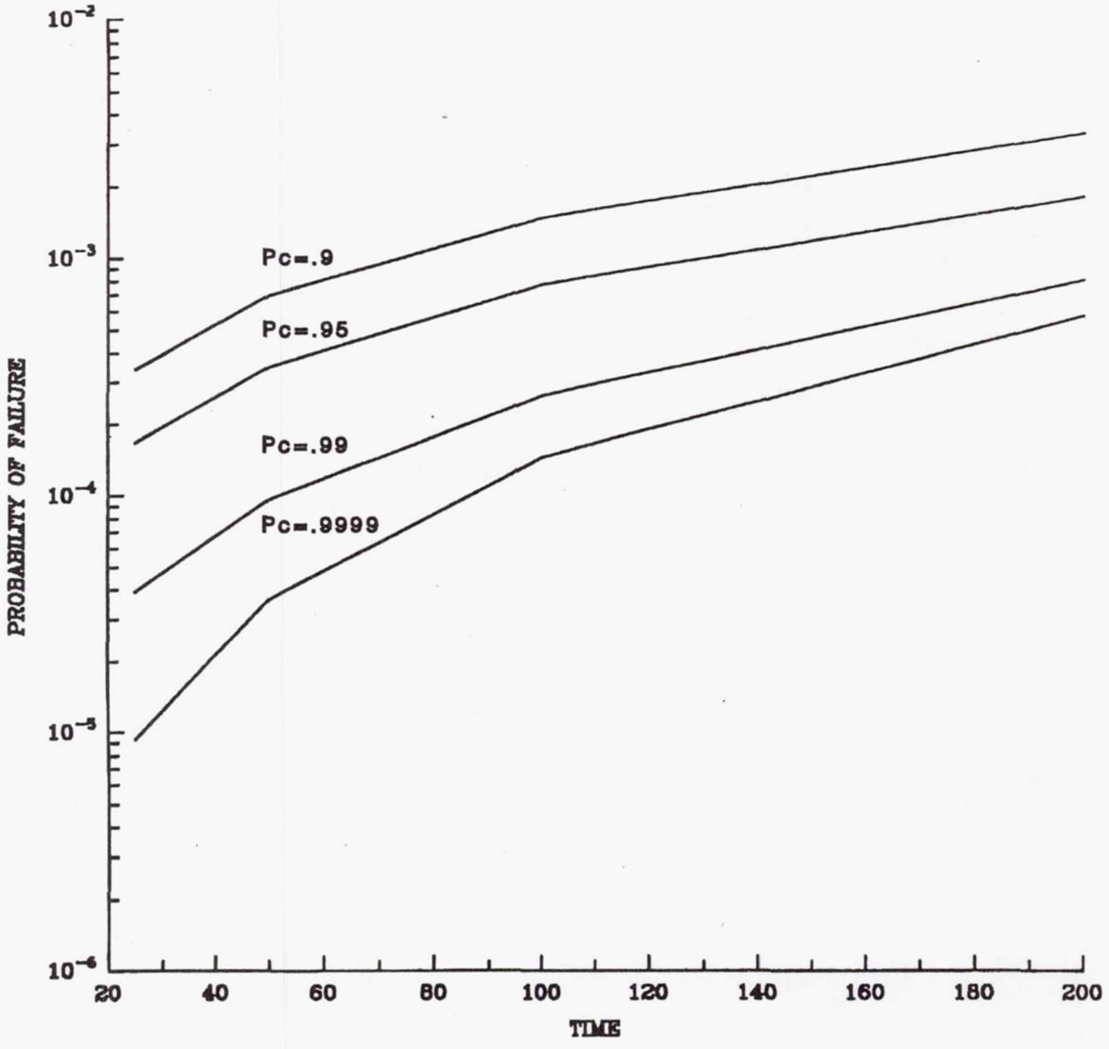


Figure 3.6. Self-Checking Pair Duplex Unreliability

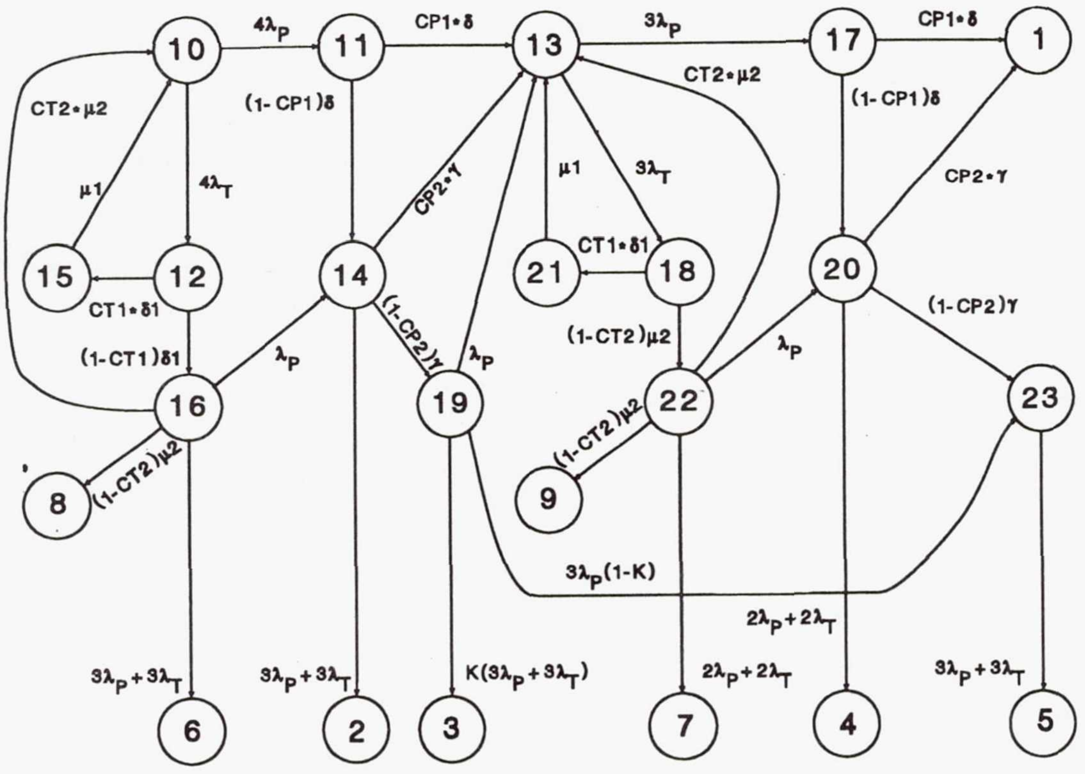


Figure 3.7. Quadruplex On-Pad Model (State descriptions are in following text)

P	= 1 if a permanent fault has occurred,
	0 otherwise
T	= 1 if a transient fault has occurred,
	0 otherwise
R	= 0 if no detection/isolation/recovery actions are in progress,
	1 if fast detection/isolation/recovery actions are in progress,
	2 if slow detection/isolation/recovery actions are in progress
F	= 0 if vehicle can be launched from this state,
	1 otherwise
UPF	= 1 if there is an undetected permanent fault present,
	0 otherwise
UTF	= 1 if there is an undetected transient fault present,
	0 otherwise

Transitions from one state to another in the model are made according to the following rates and probabilities:

λ_P	Permanent failure rate of a processor
λ_T	Transient failure rate of a processor
CP1	Probability that permanent fault is detected, isolated, and the failed processor switched out as a result of fast diagnostics
CP2	Probability that permanent fault is detected, isolated, and the failed processor switched out as a result of slow diagnostics
CT1	Probability that a transient fault is detected and isolated as a result of fast diagnostics
CT2	Probability that a transient fault is detected, isolated, and the processor recovered as a result of slow diagnostics
δ	Rate at which a permanent fault is detected, isolated, and the failed processor switched out as a result of fast diagnostics
$\delta 1$	Rate at which a transient fault is detected and isolated as a result of fast diagnostics
γ	Rate at which a permanent fault is detected, isolated, and the failed processor switched out as a result of slow diagnostics
$\mu 1$	Rate at which a processor with a transient fault is recovered as a result of fast diagnostics
$\mu 2$	Rate at which a transient fault is detected, isolated, and the processor recovered as a result of slow diagnostics
K	Percentage of permanent faults whose errors cannot be masked when another undetected fault is present in the system

The states in this on-pad model are described in the following table:

State Number	State Vector	Description
10	(4,0,0,0,0,0,0)	4 operating processors No faults
11	(4,1,0,1,0,0,0)	4 operating processors 1 permanent fault Fast FDIR in process
12	(4,0,1,1,0,0,0)	4 operating processors 1 transient fault Fast FDIR in process
13	(3,0,0,0,0,0,0)	3 operating processors 1 processor switched out due to permanent fault
14	(4,1,0,2,0,0,0)	4 operating processors 1 permanent fault Slow FDIR in process
15	(4,0,1,0,0,0,0)	4 operating processors Transient fault detected and isolated
16	(4,0,1,2,0,0,0)	4 operating processors 1 transient fault Slow FDIR in process
17	(3,1,0,1,0,0,0)	3 operating processors 1 permanent fault Fast FDIR in process
18	(3,0,1,1,0,0,0)	3 operating processors 1 transient fault Fast FDIR in process
19	(4,1,0,0,0,1,0)	4 operating processors 1 undetected permanent fault
20	(3,1,0,2,0,0,0)	3 operating processors 1 permanent fault Slow FDIR in process
21	(3,0,1,0,0,0,0)	3 operating processors Transient fault detected and isolated
22	(3,0,1,2,0,0,0)	3 operating processors 1 transient fault Slow FDIR in process
23	(3,1,0,0,0,1,0)	3 operating processors 1 undetected permanent fault
1	(2,0,0,0,0,0,0)	2 operating processors No faults Unlaunchable
2	(2,1,0,0,1,0,0)	Second failure during recovery from permanent fault
3	(2,1,0,0,1,1,0)	Second failure while undetected permanent fault exists
4	(1,1,0,0,1,0,0)	Second failure during recovery from permanent fault
5	(1,1,0,0,1,1,0)	Second failure while undetected permanent fault exists
6	(2,0,1,0,1,0,1)	Second failure during recovery from transient fault
7	(1,0,1,0,1,0,1)	Second failure during recovery from transient fault
8	(4,0,1,0,0,0,1)	Unsuccessful detection, isolation, and recovery from transient fault
9	(3,0,1,0,0,0,1)	Unsuccessful detection, isolation, and recovery from transient fault

This model was used to determine the probability of being in various states at the end of the pre-launch phase. These probabilities will be used to initialize the starting states of the launch model. Of particular interest is the probability of being in a state where a failure has occurred but has not been identified and the faulty unit has not been removed from voting. Figure 3.8 shows a portion of the model results for the undetected failure state given various effective coverage values and channel failure rates between 5 x 10^{-5} and 1 x 10^{-3}/hr at the end of 200 hours of powered operation prior to launch.

The results for pad times of 25 and 50 hours are approximately $\frac{1}{8}$th and $\frac{1}{4}$th respectively of the 200 hour results. These results were used to instantiate the state probability for the imperfect diagnostics state in the launch model.

Figure 3.9 shows the probability that a single quadruplex processing site will degrade sufficiently to prevent launch at the end of 200 hours of pad time for channel failure rates between 5 x 10^{-5} to 10^{-3}/hr.

Quadruplex Launch – A Markov model for a quadruplex architecture during the launch mission phase is illustrated in Figure 3.10. This model combines the permanent and transient processor failure rates of the on-pad model with a launch environment factor into one processor failure rate, λ_L. The model contains fault occurrence states for up to 3 faults, allowing the initial quadruplex configuration to degrade to a simplex. The fault handling mechanisms are modeled simply as a holding state which leads either to a recovered state or to system failure. Since the ability to degrade from a duplex to a simplex relies on a different fault isolation and recovery mechanism than is required for the other reconfigurations, a different recovery rate and probability are assigned to the duplex-to-simplex transitions. States 9 and 12 in Figure 3.10 represent the probability that the launch mission is begun when an undetected permanent fault is present in the system. Given that the initiation of the launch phase is contingent upon the successful completion of the on-pad phase, states 7, 10, 9, and 12 are assigned initial probabilities equal to their occupancy probabilities at the completion of the on-pad phase. These occupancy probabilities are determined from the on-pad model described previously.

The states in this model can be represented by a vector

$$S=(N,F,R,UF), \text{ where}$$

N = the number of operational processors in the state
F = 1 if a fault has occurred and detection/isolation/recovery actions are in progress
0 otherwise
R = 1 detection/isolation/recovery actions are unsuccessful or a second fault has occurred during detection/isolation/recovery

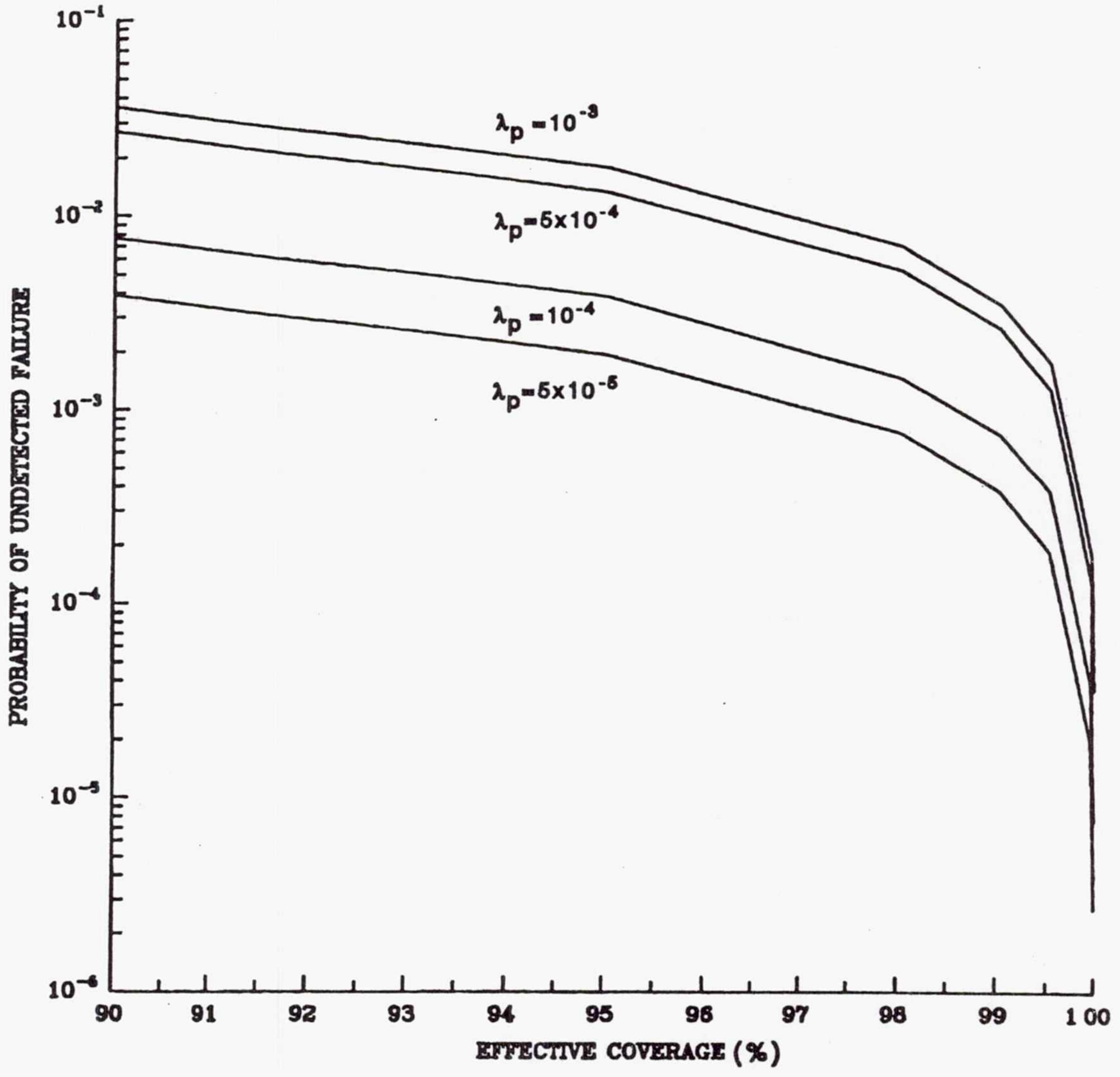


Figure 3.8. Quad On Pad

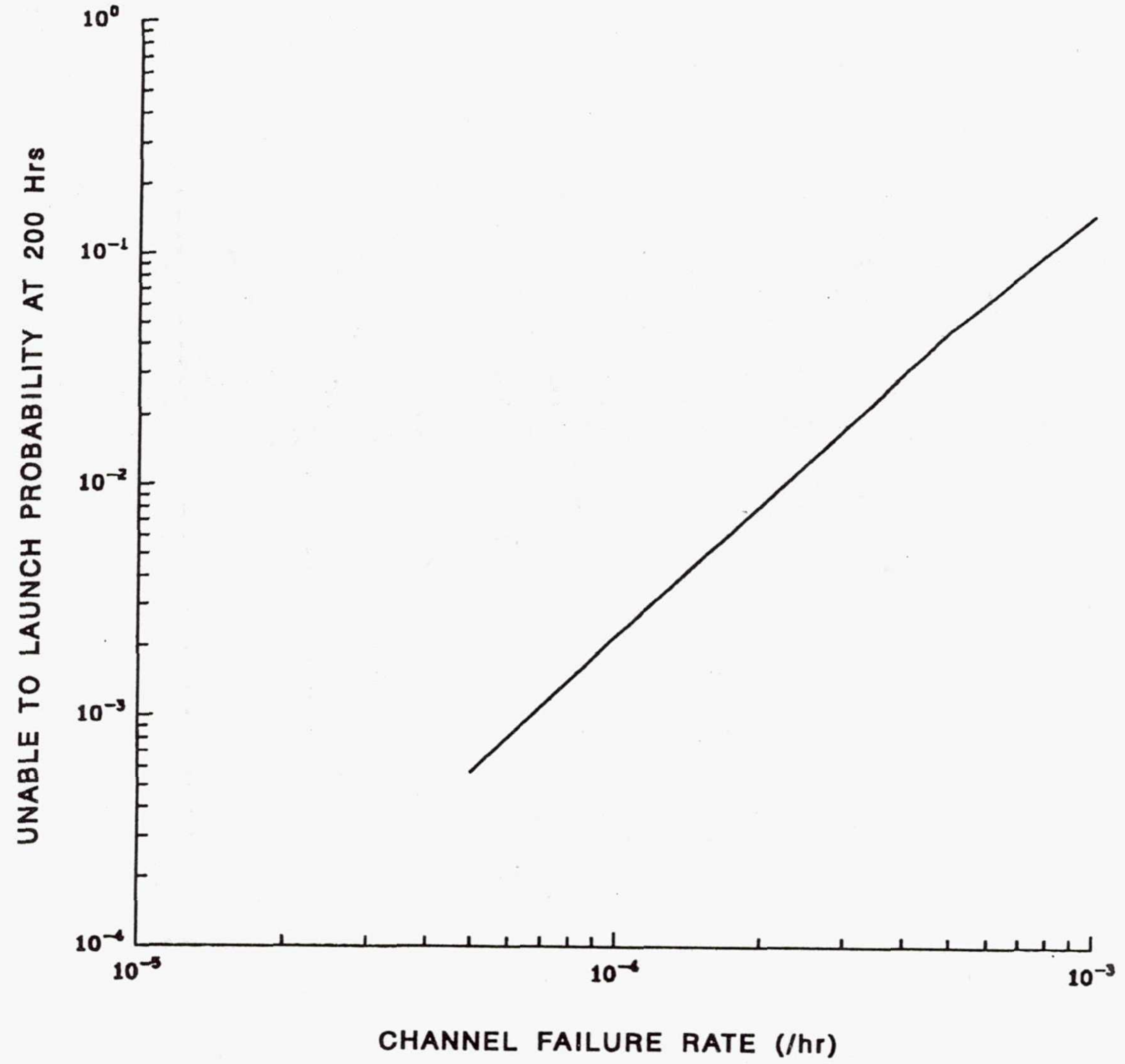


Figure 3.9. Quad Failure to Launch

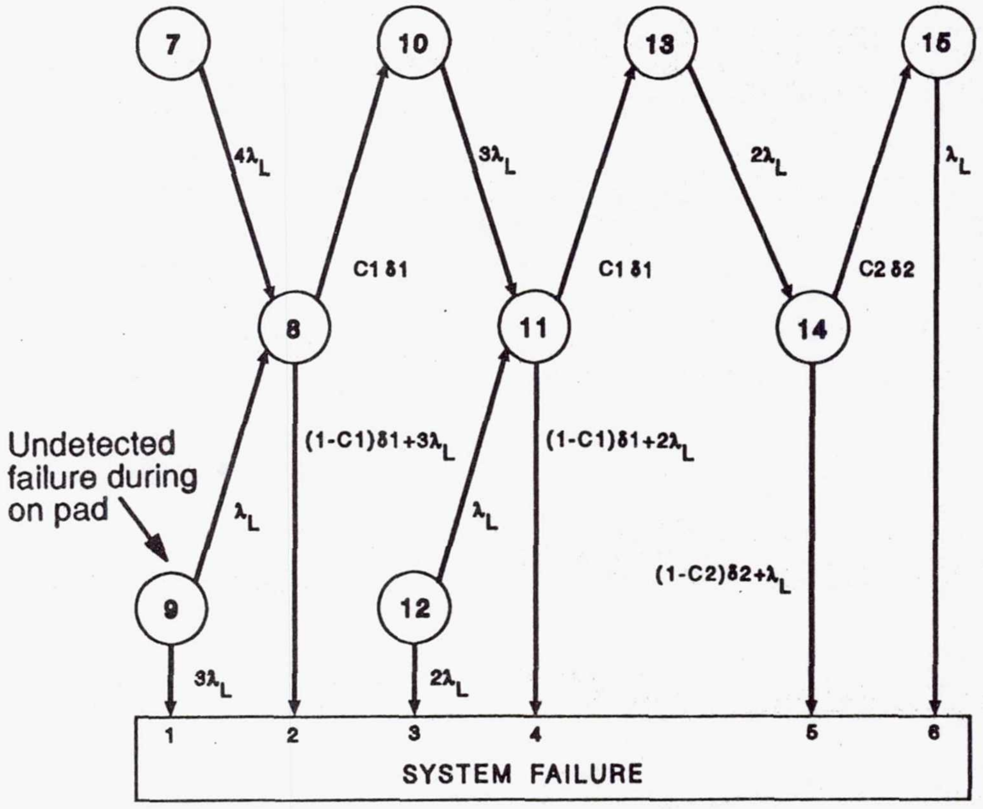


Figure 3.10. Quadruplex Launch Model (State descriptions are in following text)

0 otherwise
UF = 1 if there is an undetected fault present,
0 otherwise

Transitions from one state to another in the model are made according to the following rates and probabilities:

λ_L	Failure rate of a processor
C1	Probability that a fault is detected, isolated, and the failed processor switched out given that there are more than 2 non-failed processors
C2	Probability that a fault is detected, isolated, and the failed processor switched out given that there are only 2 non-failed processors
$\delta 1$	The rate at which a fault is detected, isolated, and the failed processor switched out given that there are more than 2 non-failed processors
$\delta 2$	The rate at which a fault is detected, isolated, and the failed processor switched out given that there are only 2 non-failed processors

The states in this launch model are described in the following table:

State Number	State Vector	Description
7	(4,0,0,0)	4 operating processors No faults
8	(4,1,0,0)	4 operating processors 1 fault FDIR in process
9	(4,0,0,1)	4 operating processors 1 undetected fault (due to on-pad failure)
10	(3,0,0,0)	3 operating processors 1 processor switched out due to fault
11	(3,1,0,0)	3 operating processors 1 fault FDIR in process
12	(3,0,0,1)	3 operating processors 1 undetected fault
13	(2,0,0,0)	2 operating processors No faults
14	(2,1,0,0)	2 operating processors 1 fault FDIR in process
15	(1,0,0,0)	1 operating processor No faults
1	(4,1,0,1)	4 operating processors a fault has occurred while an undetected fault is present
2	(4,0,1,0)	4 operating processors detection/isolation/recovery actions are unsuccessful or a second fault has occurred during detection/isolation/recovery
3	(3,1,0,1)	3 operating processors a fault has occurred while an undetected fault is present
4	(2,0,1,0)	2 operating processors detection/isolation/recovery actions are unsuccessful or a second fault has occurred during detection/isolation/recovery
5	(1,0,1,0)	1 operating processor detection/isolation/recovery actions are unsuccessful or a second fault has occurred during detection/isolation/recovery
6	(0,0,0,0)	no operating processors

The launch model for the quadruplex was examined using different permanent failure rates, different transient failure rates, different launch fault coverage factors, and different pre-launch powered avionics times.

Table 3.8 gives the probability of system failure due to imperfect pre-launch diagnostics for various failure rates, pre-launch intervals, and diagnostic coverages. Transient failures are assumed to be 10 times the permanent rate.

Regardless of the length of the pre-launch interval, pre-launch diagnostic coverages greater than 0.99 would result in unreliabilities better than 3.9 x 10^{-6} or better for all channel failure rates considered. For a 25 hour pre-launch interval, coverage of 0.9 or better result in unreliabilities better than 4.9 x 10^{-6}. There are combinations of pre-launch coverage, pre-launch time and channel failure rate which will cause the

Pre-launch Time (hrs)	Channel Failure Rate (Ground Fixed)/hr	Pre-launch Coverage			
		0.9	0.95	0.99	0.995
25	$5\text{x}10^{-5}$	$7\text{x}10^{-7}$	$3.5\text{x}10^{-8}$	$7\text{x}10^{-8}$	$3.5\text{x}10^{-7}$
	10^{-4}	$2.8\text{x}10^{-6}$	$1.4\text{x}10^{-6}$	$2.8\text{x}10^{-7}$	$1.4\text{x}10^{-7}$
	$5\text{x}10^{-4}$	$4.9\text{x}10^{-6}$	$2.5\text{x}10^{-6}$	$4.9\text{x}10^{-7}$	$2.5\text{x}10^{-7}$
50	$5\text{x}10^{-5}$	$1.4\text{x}10^{-6}$	$7\text{x}10^{-7}$	$1.4\text{x}10^{-7}$	$7\text{x}10^{-8}$
	10^{-4}	$5.5\text{x}10^{-6}$	$2.8\text{x}10^{-6}$	$5.5\text{x}10^{-7}$	$2.8\text{x}10^{-7}$
	$5\text{x}10^{-4}$	$9.8\text{x}10^{-6}$	$4.9\text{x}10^{-6}$	$9.8\text{x}10^{-7}$	$4.9\text{x}10^{-7}$
200	$5\text{x}10^{-5}$	$5.6\text{x}10^{-6}$	$2.8\text{x}10^{-6}$	$5.6\text{x}10^{-7}$	$2.8\text{x}10^{-7}$
	10^{-4}	$2.2\text{x}10^{-5}$	$1.1\text{x}10^{-5}$	$2.2\text{x}10^{-6}$	$1.1\text{x}10^{-6}$
	$5\text{x}10^{-4}$	$3.9\text{x}10^{-5}$	$2\text{x}10^{-5}$	$3.9\text{x}10^{-6}$	$2\text{x}10^{-6}$

Table 3.8. Probability of Failure Due to Imperfect Pre-launch Diagnostics

processing site unreliability to exceed 1 x 10^{-5}. It is concluded that the pre-launch diagnostic coverage needs to exceed 0.95. If the pre-launch interval is closer to 200 hours, the coverage needs to be closer to 0.99.

Table 3.9 gives the conditional probability that a processing site fails during the launch phase given that the quadruplex degraded to a triplex during the pre-launch interval and experienced two failures during launch. The duplex coverage is assumed to be 0.5 and the transient failure rate is assumed to be 10 times the permanent failure rate. Only the 200 hour pre-launch and permanent ground fixed failure rate of 5 x 10^{-4}/hour approaches an unreliability of 1 x 10^{-5}.

Channel Failure Rate (Ground Fixed)/hr	Pre-launch Time (hrs)		
	25	50	200
5x10^{-5}	1.7x10^{-9}	3.4x10^{-9}	1.2x10^{-8}
10^{-4}	1.4x10^{-8}	6.7x10^{-8}	9.6x10^{-8}
5x10^{-4}	1.7x10^{-6}	3.2x10^{-6}	8.3x10^{-6}

Table 3.9. Probability of Launch Failure For a Quad Degraded to Triplex During Pre-launch (Duplex Coverage = 0.5)

Table 3.10 gives the probability of failure due to an uncovered first failure during the launch phase for different values of coverage during the launch phase. Launch coverages of 0.8 and 0.9 result in a processing site unreliability above the 1 x 10^{-5} when the channel failure rate is 5 x 10^{-4}/hour.

Triplex – A Markov model for a triplex architecture is illustrated in Figure 3.11. This model includes fault occurrence and fault handling states for both permanent and transient faults. The fault handling mechanisms are modeled simply as a holding state which leads either to a recovered state or to system failure.

The states in this model can be represented by a vector

$$S=(N,T,P), \text{ where}$$

Pre-launch Time (hrs)	Channel Failure Rate (Ground Fixed)/hr	Launch Coverage			
		0.8	0.9	0.95	0.99
25	$5x10^{-5}$	$2.7x10^{-7}$	$1.4x10^{-7}$	$7x10^{-8}$	$1.4x10^{-8}$
	$1x10^{-4}$	$1.1x10^{-6}$	$5.5x10^{-7}$	$2.7x10^{-7}$	$5.5x10^{-8}$
	$5x10^{-4}$	$2.6x10^{-5}$	$1.3x10^{-5}$	$6.5x10^{-6}$	$1.3x10^{-6}$
50	$5x10^{-5}$	$2.7x10^{-7}$	$1.4x10^{-7}$	$7x10^{-8}$	$1.4x10^{-8}$
	$1x10^{-4}$	$1.1x10^{-6}$	$5.5x10^{-7}$	$2.7x10^{-7}$	$5.5x10^{-8}$
	$5x10^{-4}$	$2.4x10^{-5}$	$1.2x10^{-5}$	$6x10^{-6}$	$1.2x10^{-6}$
200	$5x10^{-5}$	$2.6x10^{-7}$	$1.3x10^{-7}$	$6.5x10^{-8}$	$1.3x10^{-8}$
	$1x10^{-4}$	$1x10^{-6}$	$5x10^{-7}$	$2.5x10^{-7}$	$5x10^{-8}$
	$5x10^{-4}$	$2x10^{-5}$	$1x10^{-5}$	$5x10^{-6}$	$1x10^{-6}$

Table 3.10. Probability of Failure Due to Imperfect Coverage During Launch

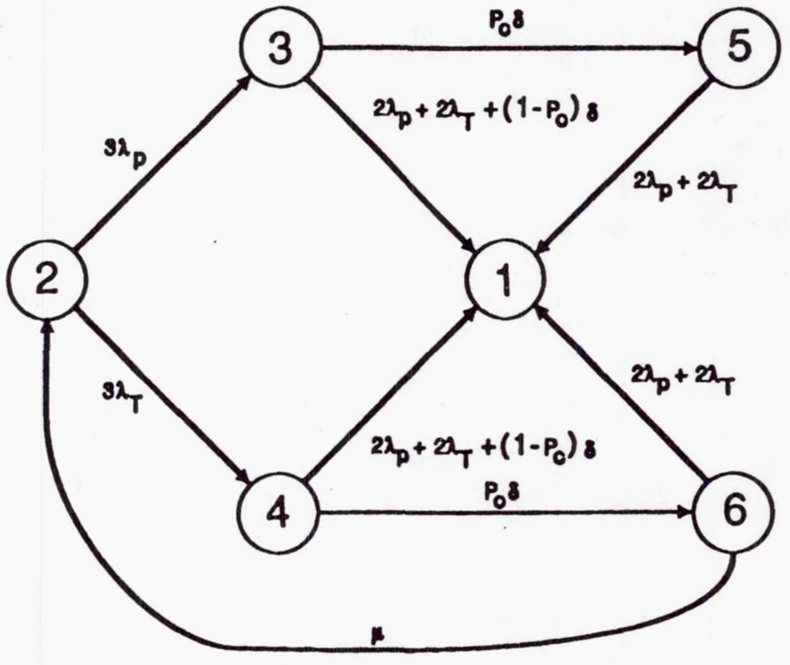

3
5
2
1
4
6
$P_o\delta$
$3\lambda_P$
$2\lambda_P + 2\lambda_T + (1-P_o)\delta$
$2\lambda_P + 2\lambda_T$
$3\lambda_T$
$2\lambda_P + 2\lambda_T$
$2\lambda_P + 2\lambda_T + (1-P_o)\delta$
$P_o\delta$
μ

Figure 3.11. Triplex Launch Unreliability

N	= the number of operational processors in the state
T	= 1 if a transient fault has occurred and detection/isolation/recovery actions are in progress, 0 otherwise
P	= 1 if a permanent fault has occurred and detection/isolation/recovery actions are in progress, 0 otherwise

Transitions from one state to another in the model are made according to the following rates and probabilities:

λ_P	Permanent processor failure rate
λ_T	Transient processor failure rate
P_c	Probability that a fault is detected, isolated, and the failed processor switched out
δ	Rate at which a fault is detected, isolated, and the failed processor switched out
μ	Rate at which a processor failed due to a transient fault is recovered

The states in this model are described in the following table:

State Number	State Vector	Description
2	(3,0,0)	3 operating processors No faults
3	(3,0,1)	3 operating processors 1 permanent fault FDIR in progress
4	(3,1,0)	3 operating processors 1 transient fault FDIR in progress
5	(2,0,0)	2 operating processors No faults
6	(2,0,0)	2 operating processor No faults
1	(0,0,0)	1 operating processor Loss of system

The recommended configuration for the General Dynamics architecture is a triplex. Assuming that the pre-launch diagnostics are perfect, the system can start the launch phase undergraded. Under this assumption the unreliability of a triplex configuration was examined for different launch fault coverages, different channel failure rates and different transient failure rates. Table 3.11 gives the approximate probability of failure for various channel failure rates for a triplex experiencing two failures during the launch phase, the second of which results in an uncovered duplex failure. Depending upon the transient failure rate scale factor, a triplex whose channel failure rates exceed 5 x 10^{-4}/hour or 1 x 10^{-3}/hour will not meet 1 x 10^{-5} unreliability requirement for a 10 minute launch phase.

Transient Factor	Channel Failure Rate/hour (Ground Fixed)	Duplex Coverage 0.5	Duplex Coverage 0.8
2	10^{-4}	$1\text{x}10^{-7}$	$4\text{x}10^{-8}$
	$5\text{x}10^{-4}$	$2.5\text{x}10^{-6}$	$1\text{x}10^{-6}$
	10^{-3}	$1\text{x}10^{-5}$	$4\text{x}10^{-6}$
10	10^{-4}	$1.4\text{x}10^{-6}$	$5.5\text{x}10^{-7}$
	$5\text{x}10^{-4}$	$3.5\text{x}10^{-5}$	$1.4\text{x}10^{-5}$
	10^{-3}	$1.4\text{x}10^{-4}$	$5.6\text{x}10^{-5}$

Table 3.11. Triplex Launch Unreliability

Multiple Core Processing Sites (Launch Phase) – Using the channel failure rates given in Figure 3.2, the reliability of a single processing site can be determined. If the General Dynamics self-checking pair (SCP) is spared in each channel, the model for the spared SCP must be used for the processor element.

Since multiple processing sites are needed for MPRAS applications, the reliability of a network of processing sites is needed. Determination of multiple site reliability depends upon the techniques used to interconnect the sites. For certain designs, the interconnection mechanism can be treated as an N modular redundant component whose failure probability contributes to system failure as an independent component and each processing site can be treated as independent N modular redundant components. At the other extreme, the interconnection is such that the channel failure rates for each processing site and for one interconnect channel must be combined to give a composite channel failure rate. This connection scheme will be referred to as direct connection in this discussion. Analysis of systems which do not fit either of these categories is often difficult due to the fact that events leading to failure are not always independent and mutually exclusive.

The analyses presented by Boeing in their MPRAS Fourth Quarterly Review assumed that the interconnection scheme is a quad modular bus whose failure can be treated independently. Their justification for this assumption is based on the use of a switching bus interface unit (BIU) to provide processing site access to the quad redundant busses. The derivation of General Dynamics MPRAS dependability goals, as presented in MPRAS Reference Vehicle Requirements, implies that the multiple processing sites are interconnected such that the core processing can be treated as a composite multistring system. That is, the channel failure rates for all sites can be added to give a composite channel failure rate. This implies that processing channels for each site are directly connected to a bus and that no cross-channel voting or bus switching is employed.

The AIPS interconnection provides for redundant connection layers or channels with voting or selection of data received on each layer. Processing channels can only transmit on one layer. Analyses which simplify and bound the reliability of this type interconnection have been presented.

If the direct connection of processing site channels without cross-channel voting or selection is assumed for the General Dynamics architecture, the processing site channel failure rates for all critical sites which are interdependent should be combined and used as the channel failure rate for the composite redundant system. Depending upon the assumption for the transient failure rate, a triplex which is the recommended configuration for the General Dynamics architecture will not meet a 10 minute launch unreliability goal of 1 x 10^{-5} when the composite ground fixed channel failure rate exceeds either 2 x 10^{-4}/hour for a transient rate factor of 10 or 10^{-3}/hour for a transient rate factor of 2.

If M identical processing sites are required, the channel failure rate for each site must not exceed $\frac{1}{M}$th of these composite rates. If the estimated failure rate for a processing channel without a spare SCP given for General Dynamics in Figure 3.2 is used, the number of processing sites will be limited to between 2 and 10 depending upon the transient multiplying factor. Once again the reliability sensitivity to the transient multiplying factor and the need to better quantify this factor is apparent. Use of a spare SCP in each site processing channel would approximately double the number of processing sites that could be supported.

For the direct connection approach, the probability of system failure for highly reliable systems tends to be proportional to M^2 for a triplex and M^3 for a quadruplex.

Assuming that the interconnection allows each site and the interconnection mechanism to fail independently as N modular redundant elements and that all processing sites are identical, the probability of system failure is given by:

$$P_{SF} = 1 - (1 - P_I)(1 - P_p)^M, \tag{3.17}$$

where:

P_{SF}	=	Probability of system failure
P_I	=	Probability that interconnection fails
P_p	=	Probability that a processing site fails
M	=	Number of processing sites

Assuming that the interconnection failure probability is much smaller than the site probability of failure, the probability of system failure is approximated by:

$$P_{SF} \simeq 1 - (1 - P_p)^M. \tag{3.18}$$

If P_p is also small, the expression can be approximated by:

$$P_{SF} \simeq MP_p - \frac{M!}{2(M-2)!}P_p^2. \tag{3.19}$$

The probability of system failure for M triplex processing sites connected in this manner, under the assumptions that $P_I \ll P_p$ and P_p is small, is proportional to M and would be a factor of M smaller than the direct connect case. For a quadruplex, it is a factor of M^2 smaller. From the reliability standpoint, architectures whose interconnection mechanisms support these latter assumptions can be more readily scaled to larger configurations (increased M) than can the direct interconnect architectures. The practical limit of this approach is the point where the unreliability of the fault-tolerant interconnection due to the voters, switches and other components is no longer much smaller than the processing site unreliability. For a given interconnect failure rate, this places a lower limit on the processing site reliability beyond which system reliability will not improve.

Analyses reported for both the Boeing and AIPS architectures indicate that their proposed interconnection reliability will be better than the reliability of a processing site.

Sparing for interconnection architectures of this type takes the form of additional processing sites which may be substituted for any failed site. The Boeing architecture has provision for this type sparing. The structure of the Boeing transducer and flight control networks permit this to be accomplished by simply activating the spare processor.

Supporting sparing in this manner is within the capability of the AIPS architecture building blocks. However, switching in a single spare so that the sensor/actuator

Table 3.12. Models for Probability that Maintenance Will be Required Prior to Launch

	Site Interconnection	
Redundancy	Direct	Independent
Triplex	$3M\lambda t$	$3M\lambda t[1 - \frac{3}{2}(M-1)\lambda t]$
Quadruplex	$6(M\lambda t)^2$	$6M(\lambda t^2)[1 - 9(M-1)(\lambda t)^2]$

I/O network or networks for all processing sites in an AIPS configuration can be serviced, may not be cost effective. The need for, or method of, sparing for MPRAS applications has not been proposed for AIPS at this point.

It is feasible that both quadruplex and triplex configurations composed of several processing sites can meet the mission reliability requirements. Interconnection mechanisms which provide for voting or bus selection are superior and more scalable than direct connection methods. While General Dynamics has indicated in some documents that a direct connect, composite, multi-channel system is their recommended configuration, some documents also indicate that processing sites can vote data passed between sites. It is not clear which option is recommended.

Multiple Processing Sites (Prelaunch) – For triplex processing sites, a failure of a redundant channel in any processing site is sufficient to require maintenance prior to launch. Similarly, two redundant channel failures in any processing site is sufficient to require maintenance for a quadruplex configuration. Table 3.12 gives models that approximate the probability of maintenance for triplex and quadruplex configurations using both direct and independent interconnection of processing sites. Equation 3.19 was used for the independent interconnection configuration.

By varying λ, M, and t over the ranges of interest, the probability that maintenance with be required can be determined. With $\lambda = 1 \times 10^{-4}$/hour, which is approximately the permanent failure rate of a processing channel in each architecture; the probability that maintenance will be required for a triplex ranges from 7.5×10^{-3} to 6×10^{-1} for triplex redundance, $1 \leq M \leq 10$, and 25 hrs $\leq t \leq$ 200 hrs. Even for the most favorable parameters, the probability that maintenance will be required could be unacceptably high

For the same parameter ranges, the directly interconnected quadruplex probability of maintenance ranges from 3.8×10^{-5} to 2.4×10^{-1} and varies as M^2t^2. If $t = 200$ hrs. and $M = 2$, the probability of required maintenance would be 0.01. The independently interconnected quadruplex probability varies as Mt^2 and ranges from 3.8×10^{-5} to 2.3×10^{-2}. A 0.01 probability results when $t = 200$ hrs and $M = 4$.

3.4. Acquisition Network Reliability

The hardware modules required to support the sensor/actuator communications function in MPRAS applications represent one half or more of the total hardware requirements. It is also the case that to date no reliability analyses of the acquisition networks have been reported by the MPRAS contractors. Based on this, it was decided that this area of design should be examined more closely. Accordingly, the reliability characteristics of this function were assessed for each architecture. Given the early design stages of the MPRAS architecture, the lack of specific application details and the breadth of MPRAS applications, a hypothetical sensor data acquisition problem was synthesized. The characteristic of this hypothetical application is simplicity and its purpose is to examine general issues related to the reliability of proposed architectures.

The only reasonably complete description of the sensors and actuators for a vehicle was provided by General Dynamics in the MPRAS Reference Vehicle Requirements. This document details a representative set of sensors for what would be considered to be a demanding application for MPRAS. The application incorporates adaptive GN&C, advanced telemetry processing, integrated Health Monitoring, and Fluids Management, as well as propulsion control and other typical avionics functions. From this representative application it can be concluded that flight-critical sensors and actuators of mixed redundancy will be distributed throughout the entire vehicle. Upwards of 3000 transducers are identified for the core and booster.

Based on the characteristics of the reference vehicle requirements, a simple sensor acquisition problem was defined. This hypothetical application was then used to examine the reliability characteristics of the candidate MPRAS architectures. In this application it was assumed that sensors are grouped into N sections of the vehicle and that all sensors have the same redundancy.

Figure 3.12 shows an acquisition network to collect the sensor data. Sensor interface units collect data from a set of sensors. Multiple busses deliver redundant sensor data to central computing resources. This network has local voting planes associated with each sensor group. Thus, if properly implemented, consistent sensor data can be provided. The network is representative of the architecture proposed by General Dynamics.

Figure 3.13 shows a network which is representative of that proposed by Boeing Aerospace. This architecture has a central voting plane. It is also presumed to be the preferred configuration for the AIPS architecture, although both configurations should be feasible with AIPS building blocks.

Consider the local voter network shown in Figure 3.12. The network is shown as triply redundant, but the quad-redundant case will also be considered. Assume that the failure to deliver at least 2 redundant inputs from any sensor group is sufficient to cause system failure. Within a sensor group, loss of 2 sensor interfaces for a triplex or 3 sensor interfaces for a quadruplex configuration results in system failure. The loss of a sufficient number of busses or the loss of any voting plane will also result in failure. The sensor interface will be taken to be any component which delivers sensor data to the voter. It is assumed that the voter outputs are coupled more or less directly to the bus. A bus link will be lost if a babbling bus transmitter cannot be disabled, if bus connections fail, or if the destination receiver and bus interface fail. Assuming that all sensor group interfaces have the same failure rate and that all voter channels have the same failure rate, the probability that a local voting triplex system is operational is given by:

$$P_S G \; = \; (1-3P_v^2+2P_v^3)^N \; [1-3P_u^2+2P_u^3]^N \; \left[(1-P_B)^3+3P_B(1-P_B)^2\frac{1}{[1+2P_v^N]}\right] \tag{3.20}$$

where: P_v	=	Probability that a voter channel fails,
P_u	=	Probability that a sensor group channel fails,
P_B	=	Probability that a bus channel fails,
N	=	Number of sensor groups.

The left most factor of this equation is the probability that at least 2 of the 3 voter channels are functional. Since there are N such components, this factor appears to the Nth power. Similarly, the second factor represents that at least 2 out of 3 sensor interface channels within a group are functional. Again, this factor appears to the Nth power. The third factor is similar to the probability that two or more of three bus channels are operational, that is,

$$\left[(1-P_B)^3+3P_B(1-P_B)^2\right] \tag{3.21}$$

However, it differs due to combinations of bus channel failures and voter failures. This factor will be referred to as an "adjusted" bus reliability factor. Similarly, the

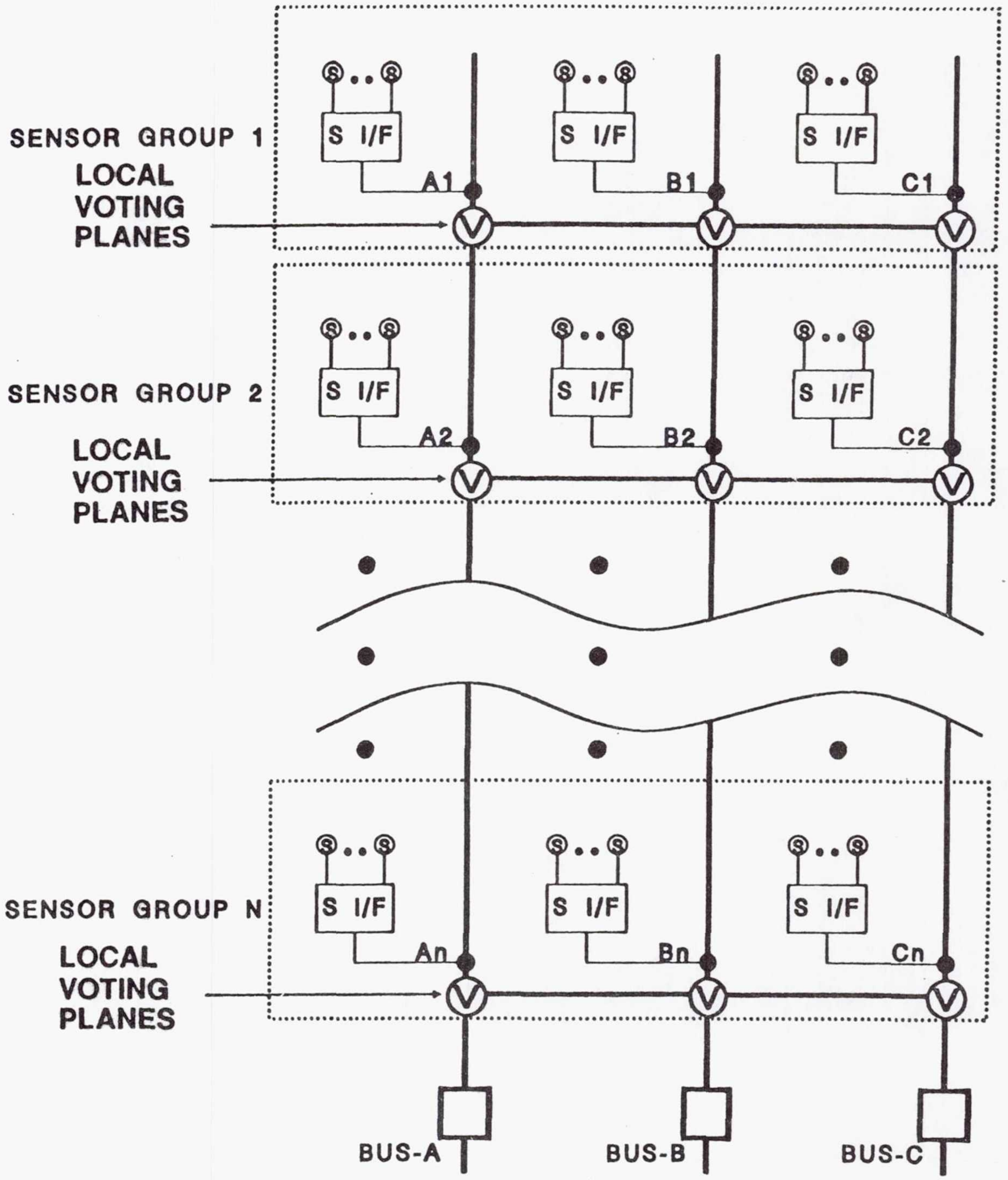


Figure 3.12. Sensor Acquisition Network with Local Voting

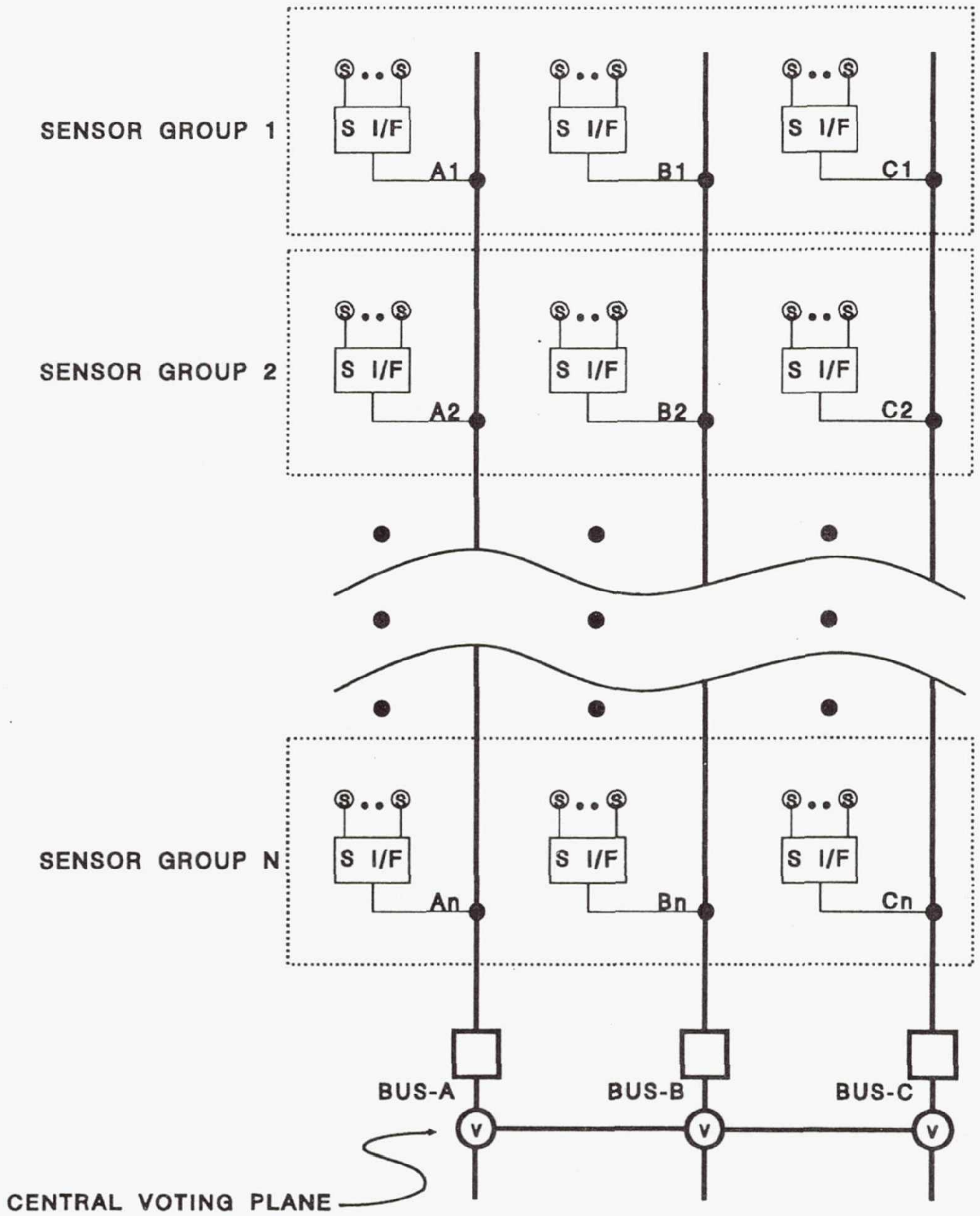


Figure 3.13. Sensor Acquisition Network with Central Voting

probability that a local voting quadruplex system is operational is given by:

$$P_SG = (1-4P_v^3+3P_v^4)^N\,(1-4P_u^3+3P_u^4)^N \left[(1-P_B)^4 + 4P_B(1-P_B)^3\frac{[1+2P_v]^N}{[1+2P_v+3P_v^2]^N} + 6P_B^2(1-P_B)^2\frac{1}{[1+2P_v+3P_v^2]^N}\right] \tag{3.22}$$

Here, the third factor is similar to the probability that two or more of four bus channels are operational, that is,

$$\left[(1-P_B)^4 + 4P_B(1-P_B)^3 + 6P_B^2(1-P_B)^2\right] \tag{3.23}$$

Consider the central voting architecture shown in Figure 3.13. Sensor interfaces are directly coupled to the bus associated with a given redundant channel. The sensor interface is likely to be less complex than that for the previous illustrated architecture since it does not have to drive a dedicated local voter. Bus failures can be taken to be the failure of bus connectors, uncontrollable failure of bus drivers at the sensor interface and the failure of the destination receiver and all support components which deliver bus data to the central voting plane. The reliability of a central voter triplex system is given by:

$$P_SG \;=\; (1-3P_v^2+2P_v^3)(1-3P_u^2+2P_u^3)^N \left[(1-P_B)^3 + 3(1-P_B)^2P_B\frac{1}{[1+2P_u]^N}\right] \tag{3.24}$$

The reliability of a central voter quadruplex system is given by:

$$P_SG = (1-4P_v^3+3P_v^4)(1-4P_u^3+3P_u^4)^N \left[(1-P_B)^4 + 4(1-P_B)^3P_B\frac{[1+2P_u]^N}{[1+2P_u+3P_u^2]^N} + 6P_B^2(1-P_B)^2\frac{1}{[1+2P_u+3P_u^2]^N}\right] \tag{3.25}$$

The form of these equations is similar to that of equations 3.20 and 3.22 in that they contain factors for the voter, the sensor interface and the bus. However, the voter factor is not to the Nth power and the bus factor is not the same as that for the local voting configuration. While the "adjusted" bus reliability factor is written in a form that is similar to that of the local case, it contains factors that depend on the sensor interface failure probability. This is due to the fact that without the intervening local voter, the loss of sensor interface units and bus channels are no longer independent.

If the probability of sensor interface unit failure is one-half, the terms in the bus factor which depend on the sensor interface unreliability (P_u) become $(\frac{1}{2})^N$ for a triplex and $(\frac{8}{11})^N$ and $(\frac{4}{11})^N$ respectively for the quadruplex. If N or P_u is increased, the factor tends to degenerate further from that of the independent redundant bus factor and tends to be influenced more by the all-busses' operational term $(1 - P_B)^K$.

A qualitative comparison of the two types of architectures suggests the potential for trade-offs between voter costs and system reliability. In the local voter case, the voter reliability factor appears to the Nth power, whereas this factor only appears once in the central voter case. For equivalent voter failure rates, this factor favors the central voting for reliability. Since the local voters do not require the higher throughput of the central voter, the potential exists for these voters to be less complex, which in turn could result in a local voter that has a lower failure rate than the central voter. The effect of this term would be weak.

If the local voter is designed so its failure rate is smaller than the sensor interface unit failure rate, the "adjusted" bus reliability factors the local voting architecture.

Tables 3.13, 3.14, and 3.15 detail, for the proposed architectures, the hardware modules and associated failure rates which are allocated to the model parameters P_u, P_v, and P_B.

For the Boeing architecture, Local Signal Conditioners are allocated to the sensor interface component. The central voter includes the Fault Tolerance module, the Processor module and a Power Supply module. The modules that are associated with bus channels include an I/O channel unit, the bus interface at the processing site and other components such as bus connectors, links, and uncontrollable failures of bus receiver/transmitter units.

The General Dynamics architecture provides for a sensor interface module that is part of a Remote Data Interface (RDI). Support for sensor interface modules is provided by the RDI. Consequently, these modules are assumed to fail at about one-half the rate associated with the Boeing Local Signal Conditioner. A bus channel is allocated the HSDB module at the receive end of the data collection bus. Other parts of the bus that can fail are connectors, links, and uncontrollable bus receiver/transmitter failures. Note also that the voter failure rate is not less than the sensor interface unit failure rate.

At this point, it is appropriate to discuss an additional feature of the General Dynamics architecture. If the number of sensors within each section of the vehicle is increased beyond the number that can be handled by a single sensor interface module in each RDI, additional sensor interface modules can be incorporated into the RDI's, and the RDI's become regional voters rather than dedicated local voters. The additional sensor interface modules can be added with less failure rate penalty than that for additional sensors in the central architectures. For the central architectures, N would double if the number of sensors in the vehicle doubled. For

Configuration: Central (Quad Recommended)

Associated Model Parameter	Hardware Module	Total Failure Rate (Ground Fixed)
Sensor I/F Channel (P_u)	Local Signal Conditioner @ 2.6 x 10^{-5}/hr	2.6 x 10^{-5}/hr
Bus Channel (P_B)	I/O channel: HSDB I/F @ 2 x 10^{-5}/hr 1773 I/F @ 1.8 x 10^{-5}/hr P.S. @ 8 x 10^{-6}/hr Processing Site: HSDB I/F @ 2 x 10^{-5}/hr Other: N Links Connectors @ N x 5 x 10^{-7}/hr Rec/Tx Babble	6.6 x 10^{-5}/hr
Voter (P_v)	Processing Site: Processor @ 2.4 x 10^{-5}/hr F.T. Module @ 2.4 x 10^{-5}/hr P.S. @ 8 x 10^{-6}/hr	5.6 x 10^{-5}/hr

Table 3.13. Boeing Sensor Network

Configuration: Local (Triplex Recommended)

Associated Model Parameter	Hardware Module	Total Failure Rate (Ground Fixed)
Sensor I/F Channel (P_u)	Sensor I/F @ 1.5 x 10^{-5}/hr	1.5 x 10^{-5}/hr
Bus Channel (P_B)	Processing Site: HSDB I/F @ 2 x 10^{-5}/hr Other: N Links Connectors @ N x 5 x 10^{-7}/hr Rec/Tx Babble (Has spare bus for each channel)	2 x 10^{-5}/hr
Voter (P_v)	RDI: Processor w/o spare @ 6 x 10^{-5}/hr w spare Local Data Link @ 2.4 x 10^{-5}/hr P.S. @ 8 x 10^{-6}/hr RDI: HSDB I/F @ 2 x 10^{-5}/hr	11.2 x 10^{-5}/hr or 5.3 x 10^{-5}/hr

Table 3.14. General Dynamics Sensor Network

Configuration: Central

Associated Model Parameter	Hardware Module	Total Failure Rate (Ground Fixed)
Sensor I/F Channel (P_u)	Device I/F unit @ 2.6 x 10^{-5}/hr I/O Node Pre-launch @ 2 x 10^{-5}/hr * launch @ 1 x 10^{-5}/hr	Pre-launch 4.6 x 10^{-5}/hr Launch 3.6 x 10^{-5}/hr
Bus Channel (P_B)	I/O Network: N Nodes Pre-launch: 1 @ 2 x 10^{-5}/hr launch: N @ 1 x 10^{-5}/hr plus 1 @ 2 x 10^{-5}/hr Launch only Connectors Rec/Tx @ N x 10^{-7}/hr FTP: IOS @ 2 x 10^{-5}/hr	Pre-launch 4 x 10^{-5}/hr Launch (4 + N) x 10^{-5}/hr Nodeless 2 x 10^{-5}/hr
Voter (P_v)	FTP Site: Processor @ 3 x 10^{-5}/hr Shared HW (Voter) @ 2.5 x 10^{-5}/hr P.S. @ 8 x 10^{-6}/hr	6.3 x 10^{-5}/hr

Table 3.15. AIPS Sensor Network

the General Dynamics modularity, the number voting planes could remain the same, and the number of sensor interface modules would double. In equations 3.20 and 3.22, the N in the voter factor and in the adjusted bus factor would be replaced by an integer equal to the number of voting planes and the unreliability of the sensor interface, P_v would be increased.

The AIPS architecture has a Device Interface Unit for sensor inputs. Since the AIPS I/O network is reconfigurable during the pre-launch period, I/O Nodes which facilitate reconfiguration will be combined with the DIU for failure purposes. That is, if an I/O Node fails prior to launch, only the associated DIU and its sensor complement are lost. During launch, the loss of an I/O Node could result in loss of the bus. It was assumed that one-half of the total I/O Node failure rate could result in failures that cause loss of the I/O network if reconfiguration was not allowed. This part of the node failure rate is allocated to the bus channel. The remaining I/O Node failure rate is allocated to the DIU.

Both during and prior to launch, the failure of the I/O Node at the receive end of this network and the I/O Sequencer in the destination FTP can in effect cause the loss of the bus channel. Prior to launch, multiple failures of connectors, links and bus Receiver/Transmitters are required to cause the loss of the bus. Otherwise, I/O network reconfiguration can configure around these sources of bus failure. The failure rate of FTP channel components where central voting occurs is allocated to the voter failure parameters of the central model.

The probability of system failure for various mission times was determined for various architecture configurations. Selected results are given in Table 3.16. The N=4 cases are equivalent to 30 quad-redundant or triple-redundant sensors in each of four sections of the vehicle or 120 multiple redundant sensor sets. The N=8 and N=12 cases are equivalent to 240 and 360 multiple redundant sensor sets, respectively. Results are given for General Dynamics in their recommended triplex configuration. Both regional and local voter, and single self-checking and spared self-checking processor cases are given. Quadruplex configurations for Boeing, AIPS, and General Dynamics are also given. The Boeing quadruplex and the AIPS quadruplex with switched I/O nodes produce essentially the same results. The AIPS quadruplex without switched I/O nodes and the General Dynamics quadruplex with spared SCPs give the most favorable results. The reliability for the General Dynamics architectures which use regional voter configurations does not vary significantly with the number of sensors. This is an indication that the reliability is dominated by the unreliability of the voter module.

If the powered pre-launch period is 200 hours, none of these configurations will meet a 10^{-5} unreliability goal. Some will meet a 10^{-4} unreliability goal. The triplex configuration will not meet a 10^{-3} goal. If the powered pre-launch period is limited

Table 3.16. Sensor Network Unreliability

Size	Time	G.D. Triplex L	G.D. Triplex A	G.D. Triplex B	Boeing Quad	AIPS Quad A	G.D. Quad B	G. D. Quad L	AIPS Quad B
N=4	25	1.3E-4	1.3E-4	3.9E-5	1.5E-7	1.6E-7	9.6E-8	9.6E-8	3.8E-8
	50	5.1E-4	5.1E-4	1.6E-5	1.2E-6	1.3E-6	7.7E-7	7.7E-7	3.1E-7
	200	7.8E-3	7.8E-3	2.4E-3	7.4E-5	7.9E-5	4.8E-5	4.8E-5	1.9E-5
N=8	25	2.6E-4	1.3E-4	4.1E-5	2.7E-7	3.0E-7	1.0E-7	1.9E-7	6.1E-8
	50	1.0E-3	5.2E-4	1.6E-4	2.2E-6	2.4E-6	8.1E-7	1.5E-6	4.8E-7
	200	1.5E-2	7.9E-3	2.5E-3	1.3E-4	1.5E-4	5.1E-5	9.4E-5	3.0E-5
N=12	25	1.3E-4	1.3E-4	4.3E-5	3.9E-7	4.4E-7	1.2E-7	2.9E-7	8.3E-8
	50	1.5E-3	5.3E-4	1.7E-4	3.1E-6	3.5E-6	9.4E-7	2.3E-6	6.6E-7
	200	2.2E-2	8.0E-3	2.6E-3	1.9E-4	2.1E-4	5.9E-5	1.4E-4	4.1E-5
		Local, No Spare SCP	Regional, No Spare SCP	Regional, Spare SCP		With Switched I/O Nodes	Regional, Spare SCP	Local, Spare SCP	Without Switched I/O Nodes

to 25 hours, most configurations except the triplex will meet a 10^{-5} unreliability goal. Note that the recommended General Dynamics triplex configuration with spare self-checking processors requires 50% more processor modules than the quadruplex versions of other architectures and does not perform as well. Note also that for the Boeing and AIPS architecture, the central voter can be spared. This approach would improve the probability associated with the bus components such as the Boeing HSDB interface and the AIPS Input/Output Sequencer. As a result the reliability of these architectures could be further improved.

While the particular configuration analyzed contains simplifying assumptions which may not be completely applicable to particular MPRAS applications, it is sufficiently representative that the results should be of concern. The topology of the networks match those proposed both by Boeing and General Dynamics and the complexity or sizes considered are well within MPRAS requirements. Since the resulting reliabilities are near or fall short of mission reliability requirements, the design of the acquisition network should be examined much more closely. This is particularly true for triplex configurations. If well-designed, the local voting approach can exhibit better reliability, scalability and performance characteristics.

3.5. Testability

Testability is the ability of an item to undergo valid. dependable functional testing and associated fault detection/isolation, within constraints of elapsed time, complexity of access, support equipment and functional procedures, and within set limits of manpower, material and other resources.

Testability underlies reliability, fault tolerance, maintainability, availability and productivity. For the ALS, MPRAS testability is a key factor which enables cost savings. The costs associated with test and maintenance for the current launch system are significant. These include the cost of extensive checkout at all stages of vehicle integration and the cost of support personnel and equipment to respond to failure prior to launch. Testability can reduce the cost of building a system. The cost of reworking a system due to an undetected fault at a particular assembly stage increases as assembly proceeds past that point. Improved testability at all assembly stages reduces both the frequency of reworks and the extent that assembly can proceed before a fault is finally detected.

The increased complexity of MPRAS avionics increase testing requirements substantially beyond that required for current launch vehicle avionics. With the large number of components in the system, it is reasonably certain that there will be several component failures during the pre-launch period. Improved testability results in better fault isolation and improved mean-time-to-repair. More importantly, improved fault diagnosis reduces the probability that the vehicle will be launched when the avionics is in a degraded state that could jeopardize a successful launch. Hence, mission reliability can be improved.

Testability features for each architecture were reviewed to determine if they had been incorporated into the design to the extent appropriate for the concept, requirements and early design stages and to determine if testability requirements are consistent with the system maintainability, reliability, availability and fault tolerance objectives.

The testability goals/requirements defined at this stage in MPRAS should include, but not be limited to, the following subjects:

1. requirement for status monitoring.
2. definition of the failure modes specified to be the basis for test design.
3. requirement for failure detection (failure coverage, failure latency) using full test resources.
4. requirement for failure detection using built-in test resources.
5. requirement for failure detection using only passive monitoring.
6. requirement for limiting false alarm rate.
7. requirement for failure localization to a subsystem/equipment using built-in test.
8. requirement for failure isolation to one or more number of modules using built-in test. The requirement may be expressed in terms of percentage of modules in a subsystem/equipment.
9. requirement for failure localization/isolation times.
10. restrictions on built-in test resources in terms of hardware size, weight and power, memory size and test time.
11. requirement for BIT hardware reliability.
12. BIT MTBF.
13. allowable down time.
14. percentage of false alarms.

15. mean fault detection time.

16. mean BIT running time and frequency.

17. maintenance skill levels.

18. system modularity.

19. test point isolation.

20. number of maintenance points and access.

21. test equipment and access.

The General Dynamics requirements establish goals for the MTTR, for fault detection and for fault isolation. Testing which supports assembly, integration and pre-flight checkout is composed of tests that are directed toward several levels of hierarchy. These include chip level, module level, subsystem level and system level. These tests are to be used in a manner that reduces the time required to test the overall system but maintain a high level of fault detection. The hierarchical breakdown reduces the testing required for complex systems.

Chip level testing is directed toward complex VLSI functions and relies upon on-chip built-in-test (BIT). For complex VLSI functions, BIT is often the only practical way to determine that a chip is fault free with a high level of confidence. Module level testing is directed toward line replaceable modules and will rely upon both on-line and off-line self-testing as well as error detection mechanisms. For example, the processor/memory proposed is to implement a self-checking pair. Module testing and error reporting is to be supported by the test and maintenance bus interface provided to each module. System level testing is supported by a health maintenance controller. This controller receives health and status reports from modules and subsystem, monitors module/subsystem self-test functions, controls system configuration for test and diagnostics, provides diagnostic capability for subsystem interconnections, monitors system communications for errors, inserts data to check error detection mechanisms and directs, monitors and diagnoses system level tests. Independent communication will be provided to each channel of the redundant paths to maintain isolation capability. Standard test interfaces are specified.

Consider the test and maintenance hardware for a triplex system which has ten engine controllers, 5 remote data units, a vehicle management processor and a guidance and navigation processor. There will be 3 system test and maintenance modules (STM) in each of these or 17x3=51 STM modules. Assume the failure rate for these modules in a ground fixed environment is $2\text{x}10^5$/hr. Also, assume that the test related hardware on each module in the system represents about 20% of the module complexity. If the average module failure rate is $2.5\text{x}10^{-5}$/hr, the failure rate of the related test and maintenance components is $5\text{x}10^{-6}$/hr. The combined failure rate for the test and maintenance hardware in a 200 module system is $200\text{x}5\text{x}10^{-6} + 51\text{x}2\text{x}10^{-5} \simeq 10^{-3}$/hr. For 25-200 hours of pre-launch operation, the probability of a failure in the test and maintenance hardware ranges between 0.025 and 0.18. Since there is a significant chance that test and maintenance hardware can fail in pre-launch operations and since the desired probability of not detecting a system fault is somewhat lower than the chance of test failure, the reliability and testability of the test and maintenance hardware appears to be a risk area for the MPRAS design.

In addition to the self-checking pair; the use of memory error detection and correction, communications error detection and correction, voting and software diagnostics to flush out latent faults are suggested for in-flight operation. The specifics of these mechanisms are not defined.

The Boeing health monitoring and BIT supports self-test, monitoring, readiness evaluation for launch, maintenance and assembly operations. Health monitoring is organized into a hierarchy of levels which are:

- Vehicle Level
- Stage Level (e.g., core or booster)
- Module Level (e.g., P/A module, payload)
- Subsystem Level (e.g., propulsion processing)
- Component Level (e.g., computer enclosure)
- Subcomponent Level (e.g., circuit board)
- Part Level (e.g., integrated circuit or sensor)

Tests for each level include internal self-test with isolation from assemblies of same or higher levels, external interconnect test with assemblies of same or higher level and external test of the assembly with BIT of higher assembly level.

The Boeing functional specifications call for an advanced, comprehensive, thorough health monitoring and BIT system. The philosophy, constraints, goals and guidelines for the design of this function have been laid out for the core processing as well as for the sensor/effector elements. Extensive descriptions of the sensor/effector failure modes and tests have been developed. Standard test interfaces are specified and each electronics enclosure will have an external test connector which will provide access to the test and maintenance port.

On-line error detecting requirements for the Boeing architecture call for error detection and correction of memory data and communications data, scrubbing of memories, and on-line diagnostics. Specific techniques and details for these mechanisms were not specified.

Extensive test and error detection mechanisms have been implemented for the AIPS proof-of-concept system. These include an on-line self-test diagnostic designed to uncover latent faults in the voter and error reporting circuits, data memory, program memory and the real-time clock. A presence test is run every processing frame prior to application processing to establish which members of a fault masking group are available. Processor exception errors that are detected include bus errors, address errors, illegal instruction errors, spurious interrupts and arithmetic traps. A watch dog timer is used to detect processors that fail to complete operation sequences. The intercomputer and input/output networks detect protocol errors, data errors and time outs. Voters are used to mask errors on intercomputer communications. A test port provides host controller access to the shared bus of the FTP. The extensive BIT that is being used for modern complex VLSI devices and the extensive BIT mechanisms needed to support thorough high coverage assembly and pre-launch testing of MPRAS avionics, including the sensor/effector acquisition distribution hardware is not implemented in the AIPS proof-of-concept. The extent to which these concepts will be proposed for MPRAS was not available for this evaluation.

Both the Boeing and General Dynamics testability and maintenance functionality and design guidelines are appropriate for the mission requirements and the stage of development. The status of the Boeing test-related requirements are at a more advanced stage than that for General Dynamics.

The BIT, self-test features and test interfaces which support assembly and pre-launch checkout for AIPS in an MPRAS application are not fully defined. The AIPS FTP and communications network self-test and error detection mechanisms which are appropriate for the in-flight and on-pad phases have been implemented and tested in the proof-of-concept system. Only the intention to use self-test diagnostics and error detection techniques have been declared for the Boeing and General Dynamics architectures. Which techniques and at what places in the architecture they are to be employed has not been detailed for either architecture. None of the architecture design information adequately addresses the reliability of the BIT and related test support hardware. At least for the Boeing and General Dynamics cases, the estimated failure rate for this is sufficient to be recognized as a risk area requiring closer attention. Not enough is known about the AIPS to determine if there is reason for concern.

4. PERFORMANCE EVALUATION

4.1. Introduction

The primary objective of the performance evaluation was to determine if the proposed MPRAS architectures are capable of handling the processing workloads projected for ALS avionics as expressed in baseline requirements. In addition, characteristics such as sensitivities to architecture parameters and workloads were considered. The evaluation takes into account the performance of the sensor/actuator data acquisition/distribution architecture as well as the basic computing resource architecture.

These performance analyses provide for the identification of strengths and weaknesses of each architecture, identification of serious design deficiencies, identification of potential development risks and critical design areas, and identification of significant differences in the designs assessed. The basis for a comprehensive performance evaluation includes specifications of hardware building blocks and their associated characteristics, specifications of software operating system characteristics, specifications of fault tolerance mechanisms and the specification of the application characteristics.

With a few exceptions, the hardware building blocks for each of the architectures have been adequately specified. The functionality and performance characteristics of the hardware fault tolerance mechanisms in the General Dynamics and Boeing architectures have not been specified sufficiently to either qualitatively or quantitatively evaluate their impact on architecture performance. Moreover, they are not sufficiently specified to assess their adequacy to support the overall fault tolerance of the architectures. For all architectures, the specification of the sensor/actuator interface is not as complete as desired. The least complete is that for the Draper Device Interface Unit.

The information of interest regarding system software architecture includes a description of control characteristics such as distributed, central or hierarchical, and descriptions of functions including task scheduling, I/O services, interrupt services, memory management, utilities, interprocessor communications services and functions related to fault tolerance. In addition, performance information such as function overheads, context switching time, expected execution time for a given system function, function response times and uncertainty in response times is of interest. The operating system software for the Draper architecture has been designed, developed and documented for a proof-of-concept system. Performance characteristics such as context switching time, fault detection isolation and recovery parameters, and certain I/O network service parameters have been measured and reported. Not reported to date are measurements for IC and I/O network message delivery times. Due to the early design stage of the Boeing and General Dynamics MPRAS architectures, specification of their operating systems design and characteristics is minimal.

An application specification which contained baseline requirements and provided a functional decomposition was requested. For each subfunction, a description of the inputs, processing, outputs and special requirements such as transport delay, jitter and process update rate is necessary. Subfunctions should be broken down into tasks and the execution sequence for these tasks should be specified. Processing workload in terms of instructions per process update and information flow between subfunctions should be characterized. The basis for the workload estimates was requested.

A good computational model specification for distributed real-time systems is essential for the design of such systems. To be consistent with "concurrent engineering" principles, good engineering practice and a methodology for the design of dependable systems, the SDIO BM/C^3 Processor and Algorithm Working Group has recommended that this model be created and delivered at the earliest milestone, the System Requirements Review (SRR). (Applicable documents a, b, and c.) Specification and modeling of the computations in an application is the starting point for the design and evaluation of computing systems. From such specifications and models, the workloads associated with the application can be characterized by temporal behavior and by function or subfunction within an application.

In the past, computational models for systems consisted of estimates of various parameters such as processing throughput, input/output data rates, and memory accesses. These estimates were often based on coarse (low-fidelity) information which was scaled up to provide adequate safety margins. While these requirements were often broken down by application subfunction they seldom provided any information regarding temporal characteristics of the workloads. Average workloads do not account for peaking factors due to multiple rate groups and transport delay requirements present in a complex real-time control system. Furthermore, average values are not adequate if a function's workload is too large to be implemented on a single processor.

While past practice is at best marginally adequate for a real-time system, it is unacceptable for complex real-time distributed systems. If an application workload must be distributed across processing resources, allowable decomposition strategies must be specified for the application. Furthermore, the communications workload generated by each decomposition strategy must be characterized. For complex real-time applications, the deadlines associated with different processes in an application can lead to variations in workload over time. In addition, specific decompositions of the workload to reside on the distributed computing resources can dictate the specific design of the fault detection, isolation and recovery processes that must be employed.

Tables 4.1, 4.2, and 4.3 summarize the computational resource requirements derived for MPRAS applications by each of three aerospace companies. The bases for these resource estimates were not documented. Computational throughput requirements are generally consistent at least to the degree of certainty expected in the early system requirements stage. Exceptions to this observation are the areas of propulsion control and telemetry processing (TT&C). There is at least a 10 to 1 discrepancy between the propulsion throughput requirements for the Martin Marietta requirements and the Boeing and General Dynamics requirements. This difference is significant since it applies to every vehicle engine and can impact overall requirements substantially. The proposed Boeing topology has a highly centralized processing core which would not be feasible if the higher propulsion throughput is required. At the very least the Boeing topology would require a large number of processing sites. This requirement should be clarified and resolved for future MPRAS effort. The telemetry processing throughput for the General Dynamics requirements is substantially greater than that called out in the other requirements. General Dynamics requirements provide for substantial data compression and transmission of vehicle data via telemetry down links to the ground.

The throughput for most functions is somewhat less than the 10 to 20 MIP processing throughput projected for MPRAS processor modules. The Martin Marietta propulsion throughput estimate of 10 MIPS, the General Dynamics telemetry processing estimate of 9.4 MIPS and the Boeing adaptive guidance and navigation processing estimate are exceptions. If any of these functions cannot reside in a single processing module, decomposition of them will require re-examination of systems communication estimates. The core processing throughput requirements for the General Dynamics computation model includes a percentage for operating system overhead. Operating system overhead for real-time systems with a range of sensor sampling rates and control loop update rates is not always well modeled by a percentage of the process computation throughput requirements. An example is the task or context switching times which require a constant number of instructions each time a task is invoked. For low frequency tasks this overhead is much lower than for high frequency tasks. Of particular concern for General Dynamics throughput estimates are the tasks that will be associated with the sampling and processing of the acoustic and vibration sensors in the propulsion instrumentation which must be sampled at 2kHz. These tasks must be invoked every 1/2 millisecond. A context switch must be completed in significantly less than

1/2 millisecond in order to maintain real-time processing.

Sensor/actuator I/O rates are broken out in the Boeing and Martin Marietta models. There is a 6 to 1 factor between these estimated I/O rates (6mbps vs 1mbps). Interfunction I/O rates for the General Dynamics and the Boeing computation models differ by a factor of 4. These differences should also be resolved before further MPRAS work proceeds. These differences are sufficient to render certain of the proposed architectural topologies infeasible.

Appendix A contains hierarchical diagrams for the MPRAS application as described by each aerospace company. The functional decompositions and relationships between decomposed functions are provided in these diagrams.

FUNCTION NAME	I/O RT (kbps)	I.F. RT (kbps)	THRPUT (MIPS)	DATA MEM (kB)	PGM MEM (kB)
Propulsion	200	726	5.8	570	(4)
Fluids	(1)	(1)	(1)	(1)	(4)
G&N	67	270	12.8	1350	(4)
Control	67	380	.4	270	(4)
TT&C	560	852	.1	215	(4)
Communications	(2)	(2)	(2)	(2)	(4)
Ground Interfaces	(2)	(2)	(2)	(2)	(4)
Range Safety	1	30	.003	3	(4)
Mission Mngmnt	0	280	.2	25	(4)
Health Monitoring	40	654	.4	2620	(4)
Instrumentation	NA	NA	NA	NA	NA
Veh-Elem Interfaces	(3)	(3)	(3)	(3)	(3)
Power	1	50	.2	70	(4)
Fault Tol Mngmnt	(4)	(4)	(4)	(4)	(4)
Miscellaneous				5000	
TOTAL	936	3242	19.9	10120	

(1) Included in Propulsion
(2) Included in TT&C
(3) Included in Control
(4) No Information
∗ Core Stage Requirements

Table 4.1. MPRAS Computational Resource Requirements – Boeing

The computational model does not provide task sequence information, does not give

FUNCTION NAME	I/O RT (kbps)	I.F. RT (kbps)	THRPUT (MIPS)	DATA MEM (kB)	PGM MEM (kB)
Propulsion	(3)	4400	6.6 (6)	20	20
Fluids	(3)	(7)	.03	1	3
G&N	(3)	(7)	1.9	160	140
Control	(3)	3200	3.2	25	100
TT&C	(3)	(7)	9.4	7	2
Communications	(1)	(1)	(1)	(1)	(1)
Ground Interfaces	(3)	4000	1	1	5
Range Safety	(3)	(7)	.6	17	36
Mission Mngmnt	(2)	(2)	(2)	(2)	(2)
Health Monitoring	(3)	(7)	9.8	3035	2254
Instrumentation	(5)	(5)	(5)	(5)	(5)
Veh-Elem Interfaces	(5)	(5)	(5)	(5)	(5)
Power	(3)	(7)	.1	5	5
Fault Tol Mngmnt	(3)	(7)	(3)	(3)	(3)
Miscellaneous (4)	(3)	(7)	2.7	7	2
TOTAL		12800	35.3	3276	2567

(1) Included in Ground Interfaces
(2) Not a separate function for General Dynamics
(3) Included in I.F. Rate
(4) Data Recording Function
(5) Not called out for Point Design
(6) For 10 engines
(7) Relatively small values
* Point Design

Table 4.2. MPRAS Computational Resource Requirements – General Dynamics

FUNCTION NAME	I/O RT (kbps)	I.F. RT (kbps)	THRPUT (MIPS)	DATA MEM (kB)	PGM MEM (kB)
Propulsion			10 (1)	40 (1)	4000 (1)
Fluids	NA	NA	NA	NA	NA
G&N	(2)	(2)	(2)	(2)	(2)
Control (2)		185	4.32	164	930
TT&C			.35	330	1650
Communications	NA	NA	NA	NA	NA
Ground Interfaces	NA	NA	NA	NA	NA
Range Safety			.004	660	2
Mission Mngmnt	NA	NA	NA	NA	NA
Health Monitoring	NA	NA	NA	NA	NA
Instrumentation	6240		.6	14	70
Veh-Elem Interfaces	NA	NA	NA	NA	NA
Power			.005	1	1
Fault Tol Mngmnt	(2)	(2)	(2)	(2)	(2)
Miscellaneous (3)	60		6	34	7
TOTAL	-	-	11+10/ENG	$\sim$ 1200	2600+4000/ENG

(1) Per Engine
(2) Includes Central Control, Staging Control, G&N and Redundancy Management
(3) Winds Ahead

Table 4.3. MPRAS Computational Resource Requirements – Martin-Marietta

workload in terms of instructions as opposed to MIPS, does not describe the basis for the model in terms of computations being implemented, I/O operations, compiler assumptions, does not include what redundancy management operations are performed, does not provide a temporal distribution of workload and does not provide a sound basis for interfunction communications.

The function (control loop) update rates, latency and jitter have been specified for each computational model. However, there is no specification for the degree of skew allowed between samples taken from different sensors for the same time index. For the fuel slosh or structural vibrations or bending measurements that could be used in an adaptive control loop, it should be expected that a high degree of time coherency would be required across spatially distributed sensors in order to best learn and adapt for vehicle dynamic parameters. A time coherency specification could impact the design for the control and synchronization of distributed sensor sampling hardware.

It is strongly recommended that the MPRAS computation model be refined and the areas where substantial differences exist between the three models described herein be resolved before further MPRAS development proceeds.

A detailed audit of the sensors and actuators required to support the application was requested. The audit should specify for each sensor/actuator the type, proposed redundancy, number of each type, number of bits, source/destination subfunction associated with the sensor/actuator, associated failure rate for each mission phase and on line test and calibration requirements. Short of a detailed audit a coarse specification that provides a gross decomposition of sensors as to appropriate number in various portions of the vehicle, the number of sensor/actuators that support a particular subfunction, the number of sensor/actuators that have a given redundancy level and the number that are time-critical or flight-critical and the approximate bandwidth of each sensor category is necessary to establish credible system requirements.

The proposed topology for the hardware elements and interconnections for each vehicle configuration specified in the baseline requirements is desirable. In addition, the allocation of subfunctions to specific hardware resources should be defined for the range of MPRAS requirements.

4.2. Sensor/Actuator Data Interprocessor Communications Performance

Background – As indicated in the previous section, the sensor/actuator I/O interface and interprocessor communications, hardware compromises more than 50%

of the total MPRAS hardware. The combined data bandwidth of the numerous sensors/actuators is typically much less than the bandwidth of the communication network used to collect/distribute these data. However, providing the capability to distribute/collect these data in a manner that is fault-tolerant and that is flexible and scalable to meet diverse requirements for a broad range of vehicle configurations is not without performance considerations. Certain of the performance analyses for MPRAS indicted that since the total sensor/actuator data bandwidth or the total interfunction communications bandwidth represented only a small fraction (less than 5%) of the system communication network bandwidth that a detailed performance analysis was not necessary. Such design and performance analysis is questioned since the sensor/actuator I/O communications performance for the MPRAS architectures is a potential risk area and limiting factor. Communications for the MPRAS sensor/actuator data collection/distribution can be defined as the delivery of information with appropriate fault tolerance and error checking measures from a sensor to a destination function (software task) or the delivery of information from a source function (software task) to an actuator. Under this definition of communications, not only is the communication network links and signaling hardware part of the communications path but the I/O hardware and software is included as well. The delays associated with the network services software places limits on the overall communications bandwidth for a distributed system. When there are numerous sources and destinations uniformly sharing the communications network, a fairly high network utilization can be maintained and the performance limitations due to network services software can have modest impact on network utilization. These performance limitations are more restrictive in networks where communications are limited to a few destinations or where communications emanate from a few sources. This is precisely the case for the sensor/actuator I/O communications network. Under these circumstances, the overall communication bandwidth can be substantially lower than the network bandwidth.

Each of the few destinations or receivers were considered to have a message processing latency. During the message processing, the receiver could not accept new messages. Network utilization and overall communications bandwidth were determined as functions of the ratio of the message time on the network to the network service software message processing time.

Background – The process of systems design requires analysis of communication network bandwidths. This analysis must consider the characteristics of the message senders, the communication network, the message receivers, and the messages themselves. Often, if the combined message traffic generated by the senders is a small percentage of the communication network's bandwidth, the network is thought to be sufficient, regardless of the receiver characteristics.

However, if the receiver is unable to process its incoming messages in a timely manner, messages may be lost. Message processing for many communication networks is a software-intensive function which is typically many times slower than the amount of time that the message transmission takes on the physical communications link. Therefore, receiver characteristics cannot be ignored when analyzing a system's communication network.

Approach – With receiver characteristics in mind, a study was conducted to examine the expected performance of a network with many senders and few receivers. The network was modeled as a bus for which each message contended. It does not model network protocols such as token passing. Although this model is a simplification of the operation of most communication networks in use today, it does demonstrate the impact of receiver bandwidths on such networks. This simplification has little effect on performance predictions when the receiver delays are large relative to cable delays coupled with the presence of only a few receivers.

The model consisted of a maximum of 32 sender nodes, each connected through 32 corresponding bus nodes to a maximum of 4 receiver nodes. Figure 4.1 is a diagram of the modeled system.

Each sender and receiver node was modeled as having its own hardware, while all bus nodes contended for the same bus hardware resource. Although there were a maximum of 32 senders and 4 receivers, only the number of components of interest were active during a given simulation. Each model node seized its hardware resource for a period of time representing the time required to send, pass or receive its data structure. This period (in seconds) was calculated by dividing data structure size (in bits) by the bandwidth (bits/second) of the seized hardware component. If messages queued up at a receiver node, the node's period was modified to reflect the time required to execute all queued messages.

The network traffic model consisted of two types of messages, node-to-node and broadcast. Node-to-node messages originated in a specific sender node and were delivered to a specific destination or receiver node. Node-to-node messages were generated so that each destination node received an equal number of messages from each send node during a given interval. Broadcast messages originated in specific send nodes and were simultaneously delivered to all designation or receive nodes. A certain percentage of the messages from each send node were broadcast messages. Node-to-node messages account for sensor data that is used by a single processing site. Whereas, broadcast messages account for sensor data that is used by all processing sites.

Parameter Variation – Two model parameters were designated independent variables to be altered to observe the impact on the utilization of the model's

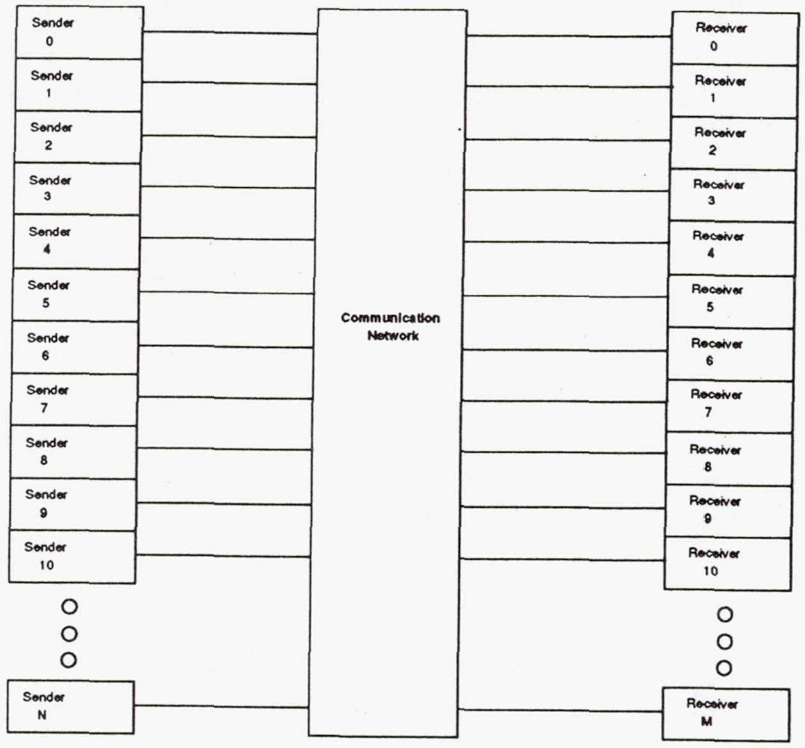


Figure 4.1. System Model

components and the system's total effective bandwidth. They were the bandwidth of each receiver and the ratio of senders to receivers. The bandwidth of each receiver was varied relative to the constant bandwidth of the network. The receiver to network bandwidths ratios were 128:1, 64:1, 32:1, 16:1, 8:1, 4:1, 2:1, 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, and 1:128. The ratio of senders to receivers were varied to include 32:4, 16:4, 8:4, 4:4, 32:2, 16:2, 8:2, 4:2, 2:2, 32:1, 16:1, 8:1, 4:1, 2:1, and 1:1. For the cases discussed in the following section, the bandwidth of each sender ws set at 1/6 of the bus bandwidth and the percentage of broadcast messages was set to a small value.

Component Utilizations Per Sender to Receiver Ratio – Sender to receiver ratios were statically established prior to each simulation run. For each run, the results showed that the utilization of each component varied with the receiver bandwidth to bus bandwidth ratio (RB:BB).

Figure 4.2 shows the percent utilizations of one sender, the bus and one receiver for the case of 1 sender and 1 receiver.

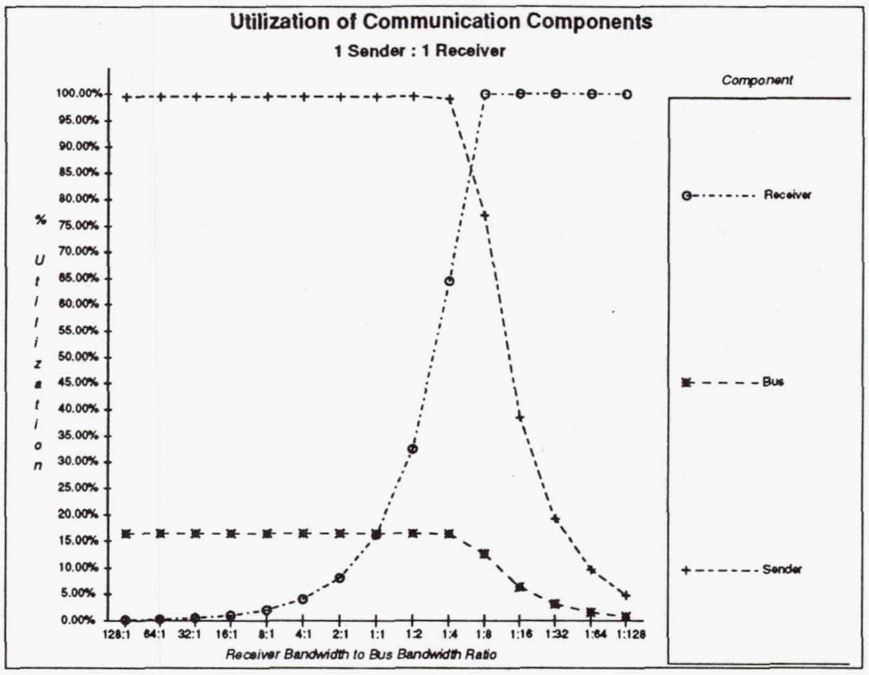


Figure 4.2. Utilization of Model Components for 1 Sender and 1 Receiver.

As the receiver bandwidth decreased relative to the bus' ability to provide data (to the 1:1 ratio), the sender was the limiting factor of the system. Both the bus and the receiver were waiting for the sender to supply messages to them. At an RB:BB of 1:1, the bandwidths of the bus and the receiver were equal, both being utilized at approximately 17 percent. Between RB:BBs of 1:4 and 1:8, the receiver's bandwidth decreased to the point where both the bus and the sender supplied messages faster than the receiver could handle. The receiver was the limiting factor of the system's effective bandwidth.

Figure 4.3 shows the percent utilizations of each of two senders, the bus and one receiver for the case of 2 senders and 1 receiver.

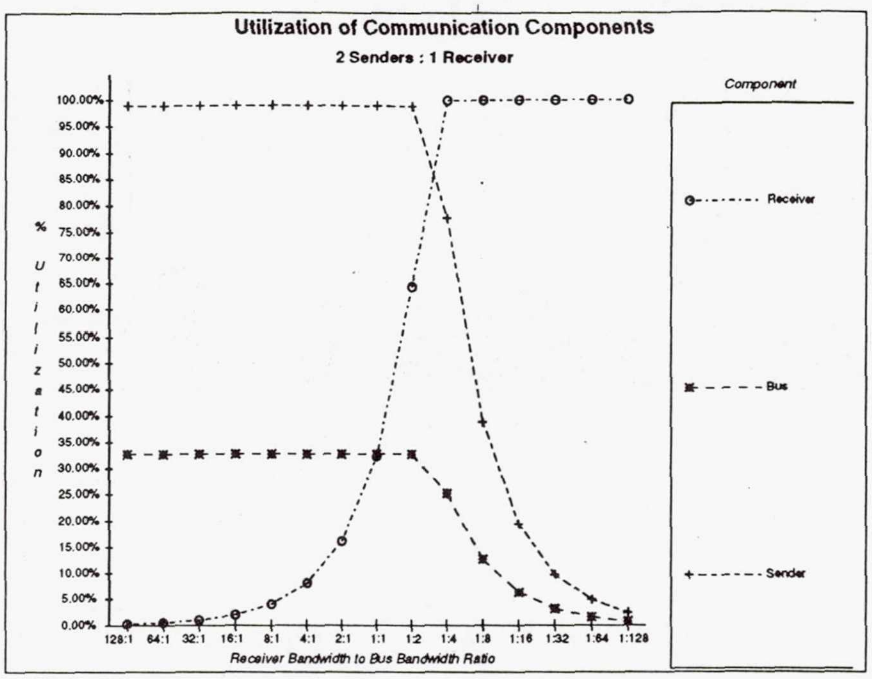


Figure 4.3. Utilization of Model Components for 2 Senders and 1 Receiver.

These results are different from the previous case. Notice that between RB:BBs of 1:2 and 1:4, the receiver's bandwidth decreased to the point of becoming the limiting factor of the system's effective bandwidth. In the prior case, this occurred between 1:4 and 1:8. The results in this instance are because there are twice as many suppliers of messages to the one receiver. Twice the number of messages also produced an increased utilization of the bus from 17 to 33 percent.

This crossover point continues to move to the left on the RB:BB scale until the S:R becomes 8:1 (Figure 4.4).

Here, the bus was the limiting factor from an RB:BBs 128:1 through 2:1. At the RB:BB of 1:1, the utilization of the bus and the receiver were the same and from RB:BBs 1:2 through 1:128 the receiver was the limiting factor.

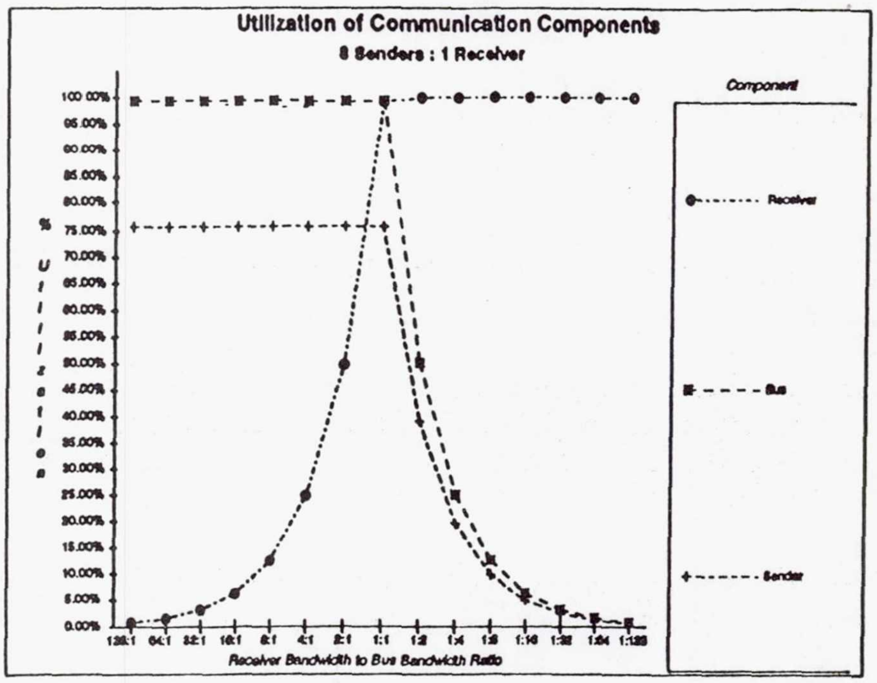


Figure 4.4. Utilization of Model Components for 8 Senders and 1 Receiver.

As the number of senders was increased while the number of receivers was kept at 1, the bus limited the system bandwidth until the RB:BB of 1:1, when the receiver became the limiting system component.

When the number of receivers was doubled to two, the crossover point of the utilization curves of the bus and each receiver became the RB:BB of 1:2. This can be explained by the fact that each receiver was receiving half the number of messages going through the bus. By comparing Figures 4.2 and 4.5, one can see that the sender and receiver utilization curves are at the same RB:BB location. The crossover occurs at this point whenever the number of senders equals the number of receivers.

Similar to the single receiver case, as the number of senders was increased and the number of receivers held constant, the bus' utilization increased, becoming the limiting factor until its bandwidth equaled the combined bandwidths of all receivers. At that point, as above, the utilization of the receiver became dominant. This can be seen in Figure 4.6.

Component Utilizations Across Sender to Receiver Ratios – The impact of the S:R can be seen by examining the utilization on one component across numerous S:Rs, such as that of the communication network as seen in Figure 4.7.

C-2

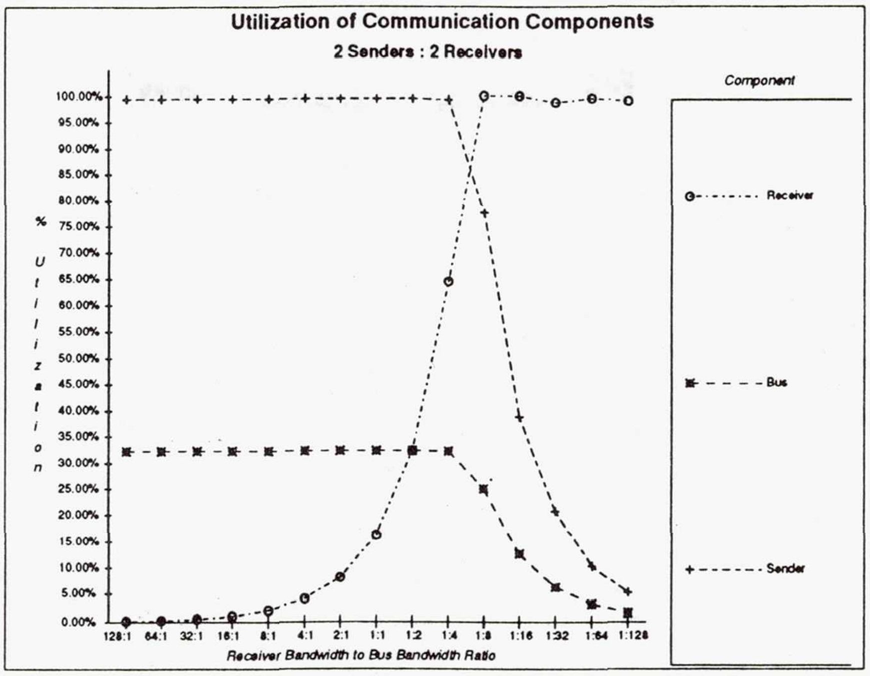


Figure 4.5. Utilization of Model Components for 2 Senders and 2 Receivers.

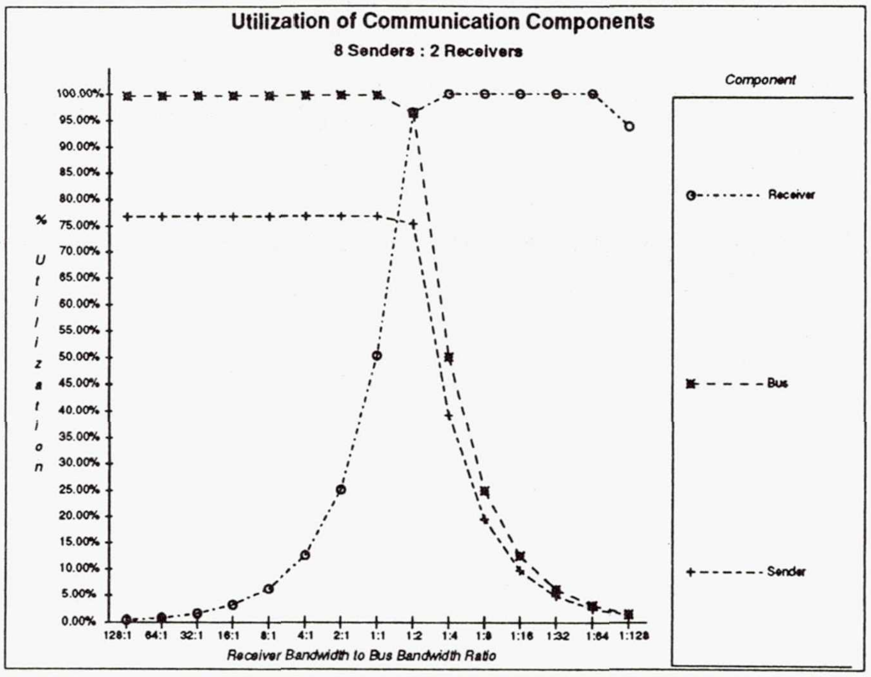


Figure 4.6. Utilization of Model Components for 8 Senders and 2 Receiver.

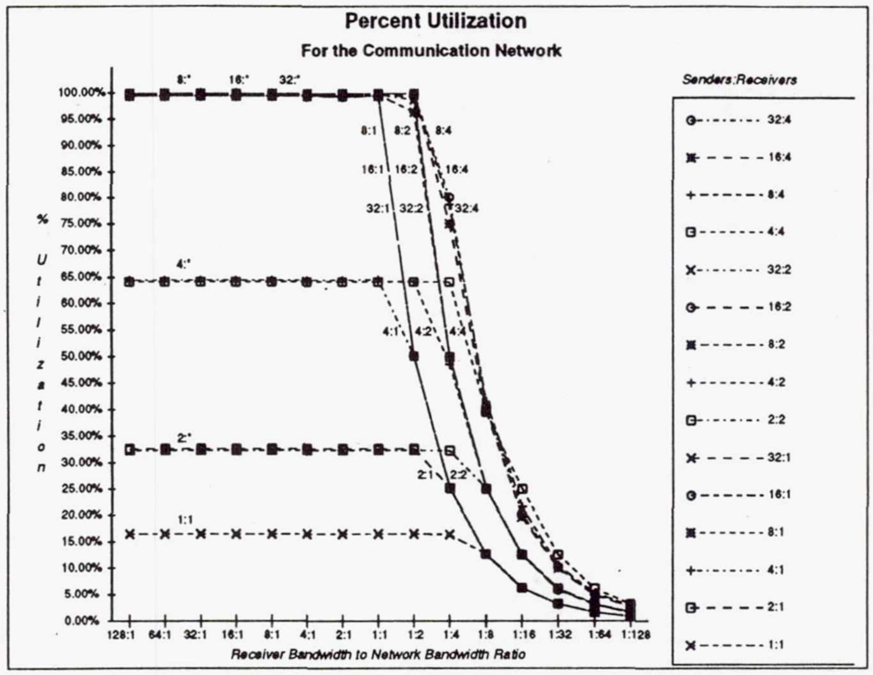


Figure 4.7. Utilization of Communication Network Across Sender to Receiver Ratios.

For cases where the number of senders was less than 8, the dominant component utilization for RB:BBs of less than or equal to 1:1 is that of the sender. For the same RB:BB range, when the number of senders is 8 or greater, the network's utilization is dominant. As discussed above, an increase in the number of receivers causes the knee of the network's utilization to shift to the right on the RB:BB scale. The corresponding curve for the utilization of each receiver is shown in Figure 4.8.

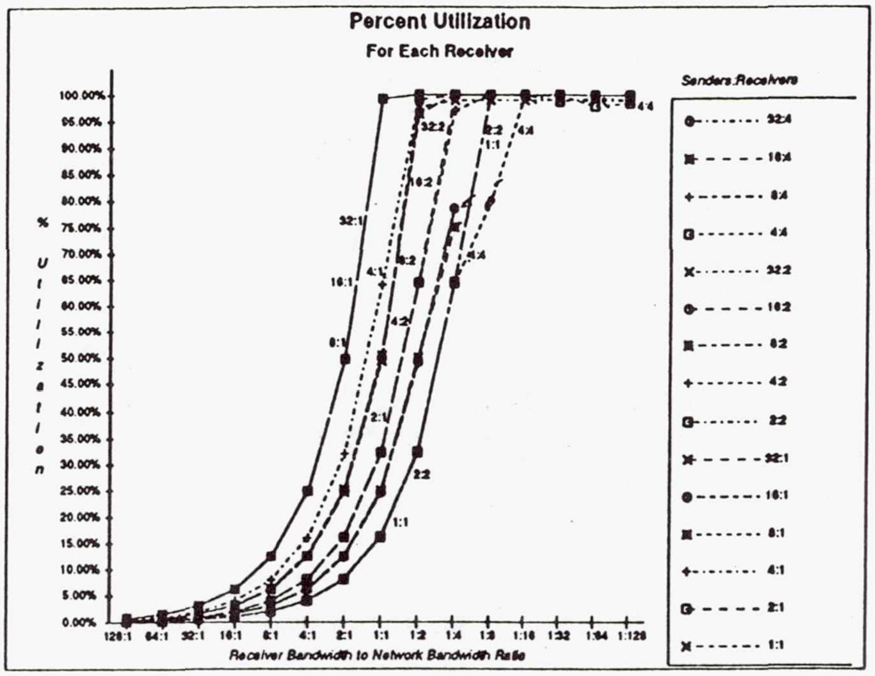


Figure 4.8. Utilization of Each Receiver Across Sender to Receiver Ratios.

Effective System Bandwidth – The results of this study indicates that the effective bandwidth of the network is determined by the combined bandwidth of all receivers and not by the bandwidth of the bus alone. As the combined bandwidth of the receivers diminished, so did the effective bandwidth of the system as a whole. The system's bandwidth relative to that of the network alone is shown in Figure 4.9.

The system's effective bandwidth was calculated by dividing the total number of bits received by the total simulation time. When the receiver message processing delays become large relative to the physical communication link message transmission time coupled with the presence of only a few receivers, link bandwidth utilization decreases. At ratios of 50:1 to 100:1, utilization is severely limited by the receiver message processing delay.

In addition, to this model, measurements of the message processing time for a

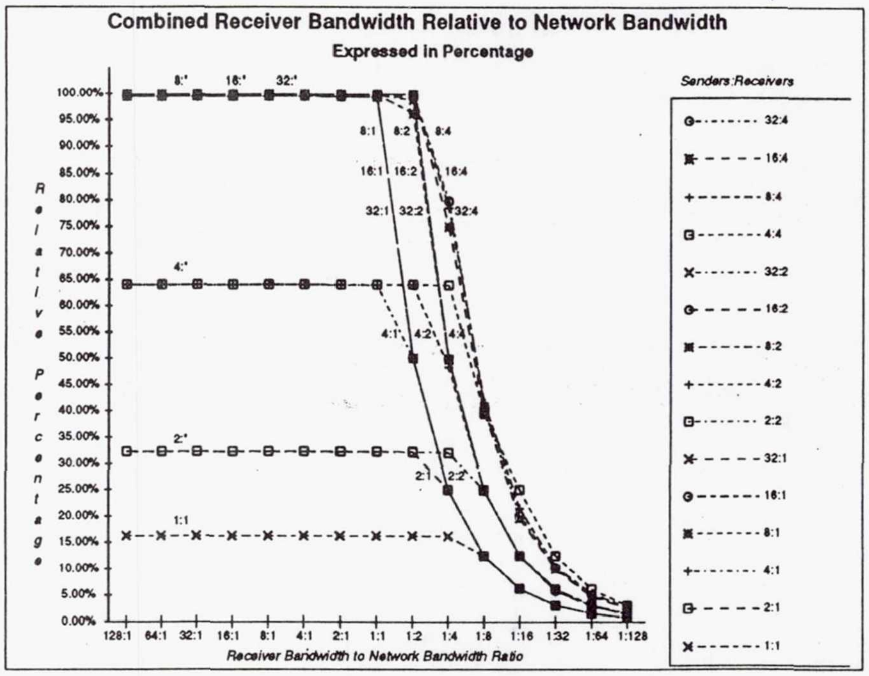


Figure 4.9. Effective System Bandwidth Across Sender to Receiver Ratios.

general purpose communication network consisting of Ethernet system coupled with DECNET and VAXELN network service software were made as a reference point. These measurements indicate a message processing time to message cable time ratio of about 100:1 for the 10Mbits/second cable and 1 Mip processors. RTI's experience with other network communications indicates that this is a not an uncommon ratio. Depending on the number of destinations or receivers, the model indicates that the maximum network utilization that can be expected would be in the 1 to 10 percent utilization range with such ratios. Good engineering practice dictates that network service message processing delays due to normal message assembly and delivery and due to error checking and fault tolerance related processing, should be analyzed and predicted to determine the limitations of the architecture. The communications models and measurements referred to in this paragraph are detailed in Appendix B.

AIPS I/O and Intercomputer Communications – Figure 4.10 shows the hardware elements associated with an AIPS FTP processing channel. A processing channel consists of a computational processor (CP), an input/output processor (IOP), a shared memory, a data exchange and voter unit, a dual ported memory (DPM), an Input/Output sequencer (IOS), an intercomputer interface sequencer (ICIS), an I/O network and an intercomputer communications network. Circuit switched I/O network nodes and links establish communications from the IOS and the DIU's. Spare nodes and links are provided so that communications to all DIU's can be maintained via reconfiguration in the event of network failures. I/O network system services software provides for communications between user tasks and the sensors and actuators connected to the I/O networks in an AIPS system. In addition to the message processing and delivery functions required for I/O communications, I/O network services provides for fault detection and isolation capability for the I/O hardware elements, reconfigures the I/O network hardware as dictated by I/O hardware failures and provides for distribution of consistent input data to all non-faulted redundant channels.

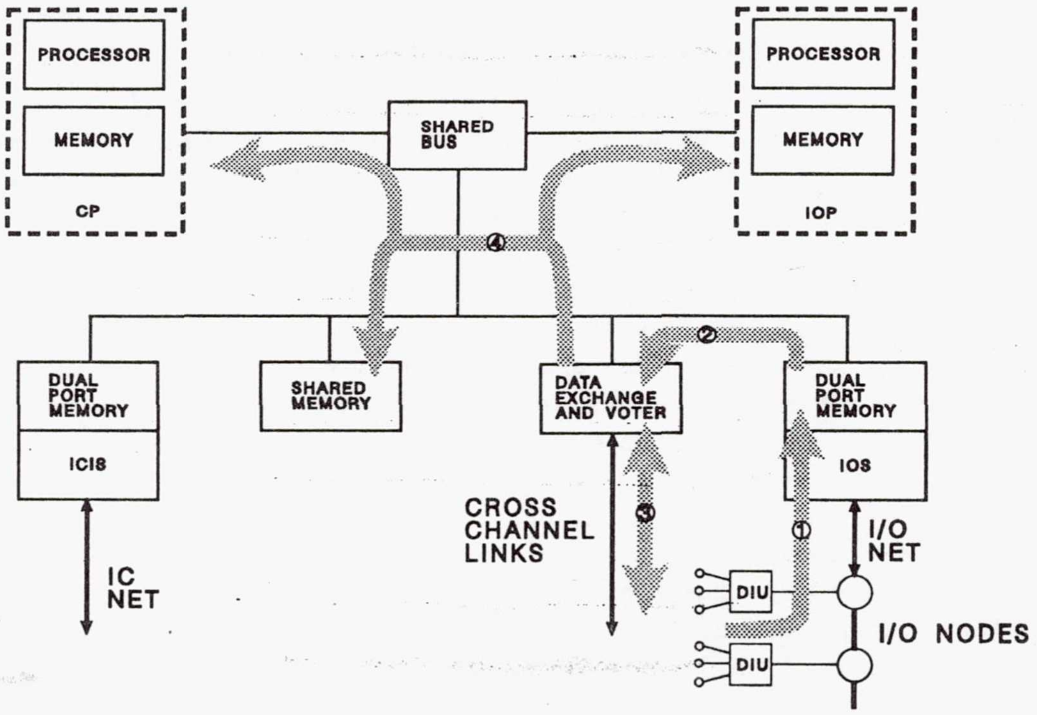


Figure 4.10. FTP Architecture and I/O Data Flow

An application task running in the CP issues an I/O request through the I/O services resident in the CP. The I/O services resident in the IOP receives the I/O request and issues a command to the IOS to activate a programmed chain of I/O network transactions which are associated with the specific I/O request. The IOS then issues commands to specific DIU's on the I/O network. For each command the selected DIU responds with a message which is processed in the IOS interface and stored in the DPM. Typically, the message contents are distributed to other redundant channels via the data exchange unit and are delivered to the shared memory. The IOP processes the received messages and performs error processing. The message contents are then delivered to the application task in the CP. Typically, only a single command is required by the IOS to carry out a complex set of I/O operations. The IOP is relieved of much of the detailed control necessary to implement the I/O transfers.

Even with the mechanization of the detailed I/O control via the IOS and programmed chains, AIPS I/O operations are software intensive. Maintaining general and flexible I/O services while providing high coverage fault detection and network reconfiguration capabilities leads to I/O throughput that is much less than the I/O network bandwidth. Effective I/O bandwidth has been one of the primary performance limiting factors of the current FTP design.

Measurements of I/O request processing times for the AIPS proof-of-concept system have indicated that the message processing time exceeds the I/O network cable time by factors of 25 to 50. On an average, in excess of 100 IOP instructions are required for each sample acquired via the I/O network. Further, to meet the latency (2 to 5 milliseconds) and control loop update rates (100Hz) required for MPRAS, the I/O request processing time must be reduced by a factor of 10 to 20. Significant I/O speedup cannot be achieved by increasing the bandwidth of the I/O network. Faster data exchange and voting hardware, higher throughput processors (CP and IOP), use of a more efficient Ada compiler and more efficient I/O service software are candidate areas to provide the necessary speedup. The AIPS processors expected to be used for MPRAS should provide at least a factor of 5 to 10 speedup over the AIPS proof-of-concept model. Additional, speedup beyond that of the processors will likely be required to meet the MPRAS requirements. Only after the I/O request processing time has been reduced by the factors indicated would it be worthwhile to increase the I/O network bandwidth by a factor of 5 to 10 and to speed-up the voter from the 2.5 to 5 microseconds per word of the current proof-of-concept model. Note that speed-up of the voter can be limited by the amount of time skew permitted between channels.

Due to I/O overhead, the effective I/O bandwidth which accounts for both the I/O network bandwidth and the I/O request processing time depends on the number of sensors that can be read in a single I/O request which in turn is application dependent. Based on the measured I/O request times reported, the effective I/O bandwidth for a flight-control application was between $1.5\text{x}10^4$ and $3\text{x}10^4$ bits/second. The actual I/O network bandwidth was $2\text{x}10^6$ bits/second. Only about 1% of the actual network bandwidth could be used. If the I/O request processing time is reduced by a factor of 20, the effective I/O bandwidth could be increased to about $5x10^5$ bits/second. Since the combined bandwidth of all MPRAS sensors and actuators is between $1x10^6$ bits/second and $6x10^6$ bits/second, an effective bandwidth of $5x10^5$ bits/second for a single I/O network should be sufficient for AIPS to meet MPRAS requirements. Figure 4.11 summarizes the structure of the AIPS I/O data delivery path.

The AIPS intercomputer network provides communications between processing sites. The ICIS shown in Figure 4.10 provides an interface to a layered (redundant) intercomputer data network and controls data transfers between the IC network and the processing channel. Each redundant channel of the processing site can transmit on a layer of the network and receives data from all layers of the network. Received data can be selected from any layer or can be voted across all layers. Circuit-switched IC nodes and data links provide communications between processing sites and in the event of network failures can be reconfigured to maintain communications.

The intercomputer system services provides intercomputer communication service for user tasks, provides a mechanism for maintaining time across distributed processing sites, and manages the fault detection, isolation and reconfiguration for each layer of the IC network. The IC communication services architecture is designed in a layered approach similar to the proposed Open Systems Interconnection protocols. The layers are:

1. the physical layer
2. the data link layer
3. the network layer
4. the transport layer
5. the session layer
6. the presentation layer
7. the process or application layer

The ICIS provides the mechanism for controlling the IC data communications via high-level commands from the IOP. Even with these ICIS features the IC system services is software intensive. In the same areas it is more complex than the I/O network services. At present the IC message delivery times for the proof-of-concept system have not been characterized. It should be expected that those delivery times will be of the same order as the I/O request processing times for the I/O network

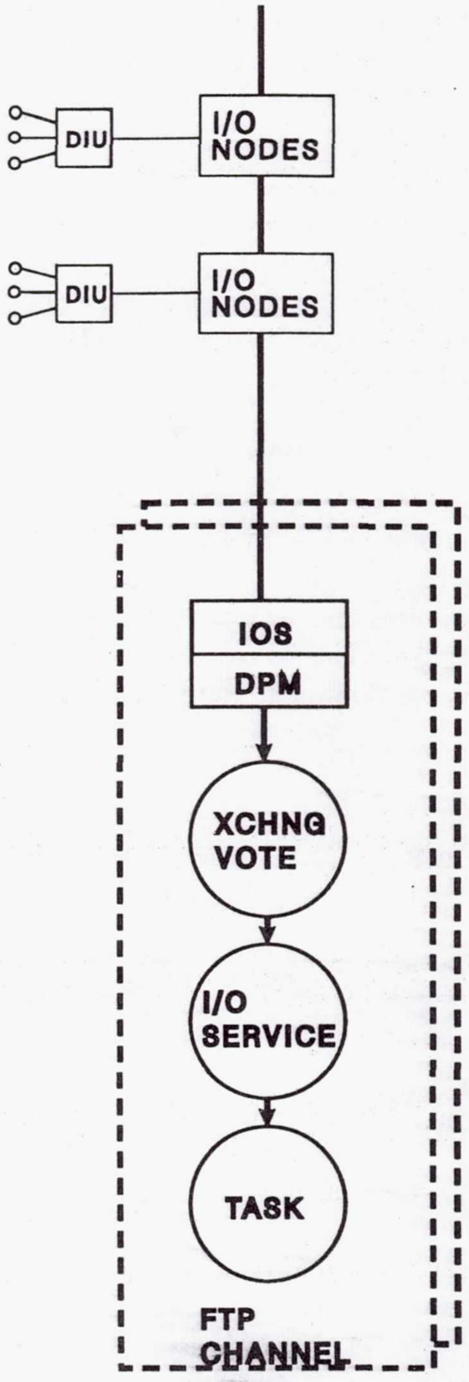


Figure 4.11. FTP I/O Structure

services. In fact, the delay components due to use of the IC network will increase due to the time required to contend for network access. Interfunction communication requirements for MPRAS are as high as 12.8x10^6 bits/second combined and range from negligible to about 3x10^6 bits/second for any single function. If functions are allocated to processing sites such that these larger interfunction communications must take place across the IC network, IC communication requirements will be demanding. To meet these requirements, the proof-of-concept IC bandwidth must be increased substantially from the current 2x10^6 bits/second.

In addition, to maintain sufficient effective IC bandwidth the IC message delivery times must be reduced from those that will be realized in the proof-of-concept system. Message delivery times of a few microseconds per byte instead of tens or hundreds of microseconds per byte will be required. If the higher interfunction requirements are valid and if the functions with the larger requirements must make use of the IC network, the IC message delivery time will be one of the highest performance risk areas for AIPS applied to MPRAS.

Network reconfiguration times for both the IC and I/O networks will not be critical for MPRAS since reconfiguration will only be allowed during the pre-launch phase when sufficient time is available for reconfiguration.

Boeing I/O and Intercomputer Communications – Figure 4.12 diagrams the hardware modules associated with a processing channel in the General Dynamics architecture. A channel is comprised of a local signal conditioner, an I/O channel module, a system bus module, a fault tolerance module, an I/O interface module, a processor module, a memory module, an output module, a MIL-STD 1773 flight control bus, a MIL-STD 1773 transducer network bus and an HSDB system bus. The software that provides for communications between application tasks and the sensor/actuator network and manages network failures has not been specified for the Boeing architecture.

Two distinct types of sensor/actuator I/O has been defined for the Boeing architecture, time critical and non-time critical. Time critical I/O is handled using the MIL-STD 1773 flight control bus. Non-time critical I/O is handled using the transducer network and the HSDB system bus. Figure 4.12 shows a conjectured sequence for time critical I/O data flow via the flight control bus. Sensor data from a local signal conditioner is transferred to an I/O interface module memory via the flight control bus. This transfer requires I/O interface module software to check for data errors and to control delivery of the data to a desired destination. It is assumed that in order to have consistent sensor data in all redundant processing channels the fault tolerance module will be used to distribute and vote sensor data across the redundant channels. This is believed to be within the intended use of the

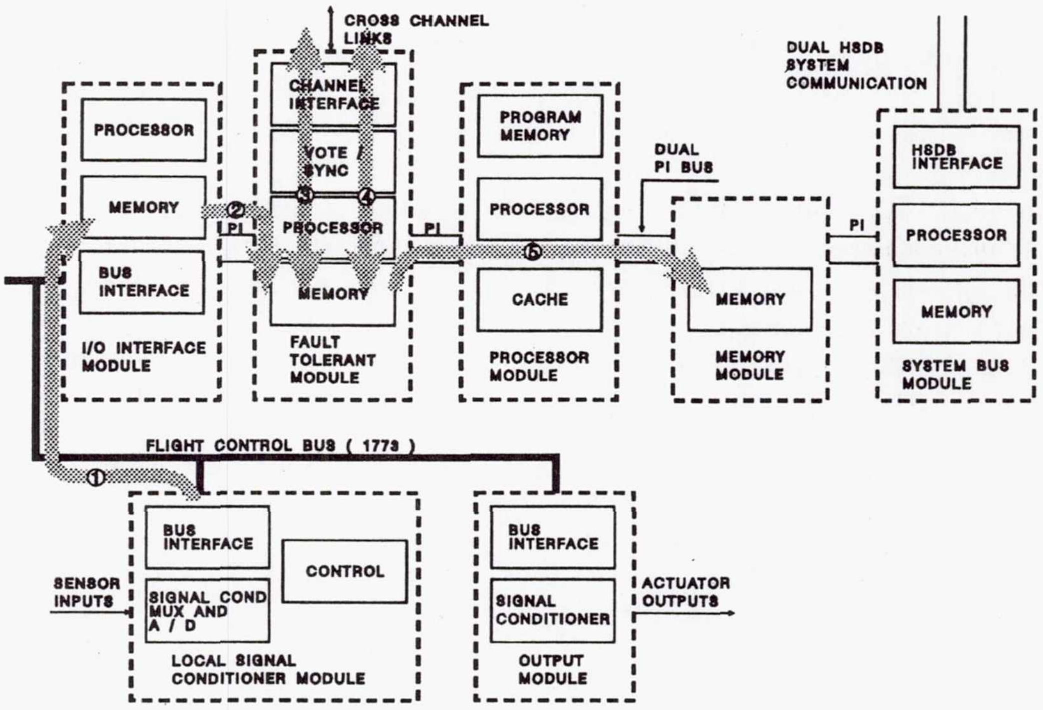


Figure 4.12. Boeing Architecture and Time Critical I/O Data Flow

fault tolerance module. However, the design for and use of this module has not been clearly specified for the Boeing architecture. Following this assumption, data would be transferred to the fault tolerance module via the PI bus. This data would be distributed to the other redundant channels via cross channel data links. Similarly, data from redundant sensors acquired by the other redundant channels would have to be distributed to all channels as well. Presumably, errors occurring during these sensor data distribution processes would be detected, processed and reported by software in the fault tolerance module. In addition, the redundant sensor data could be processed to obtain a single set of sensor data which in turn could be distributed to all channels via the cross channel links and voted by the fault tolerance module. Finally, either the redundant sensor data sets or a single voted sensor data set would be delivered to the requesting application task executing in the processor module via designated locations in processor memory. I/O service software in the processor module would be required to manage this process.

Figure 4.13 illustrates a conjectured data flow sequence for non-time critical I/O for the Boeing architecture. It differs from time critical I/O in that sensor data is transferred via the transducer network, an I/O channel module and a system bus module. (The switched BIU function shown is expected to reside on the fault tolerance module.) I/O software to direct these transfers and to detect and report errors would be required in the appropriate modules. Further, the HSDB module software would have to manage the token ring protocol of the HSDB and any required network flow control. Sensor data received in the system bus module would then be passed to the fault tolerance module where the process would proceed as previously conjectured. Figure 4.14 summarizes the structure of the Boeing non-critical I/O data network.

The message delivery time associated with sensor data would be the sum of the times required to complete each step in the process. In the worst case, the time between the delivery of one sensor data message and the next sensor data message would be the same as message delivery time. The communications bandwidth realized would be determined by this message delivery time as opposed to the bandwidth of the data transfer busses. If the message delivery process is designed so that a new message can be started as soon as a given hardware element such as the I/O interface module has completed its operations on the previous message, the time between sensor data messages will be reduced from the sum of the times required to complete all steps in the delivery process to the longest time that a single hardware module is used in the delivery process. In this case, message delivery would be designed as a pipelined sequence of steps.

Components of the overall message delivery time for the data flow sequence described above include:

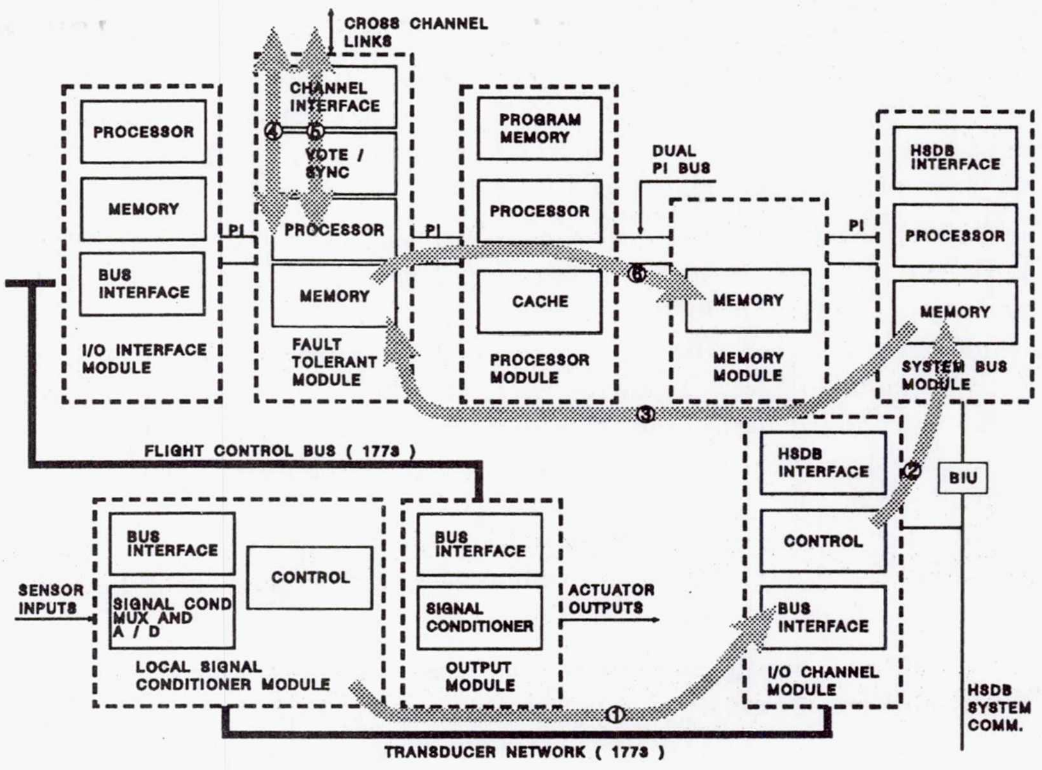


Figure 4.13. Boeing Non-Time Critical I/O Data Flow

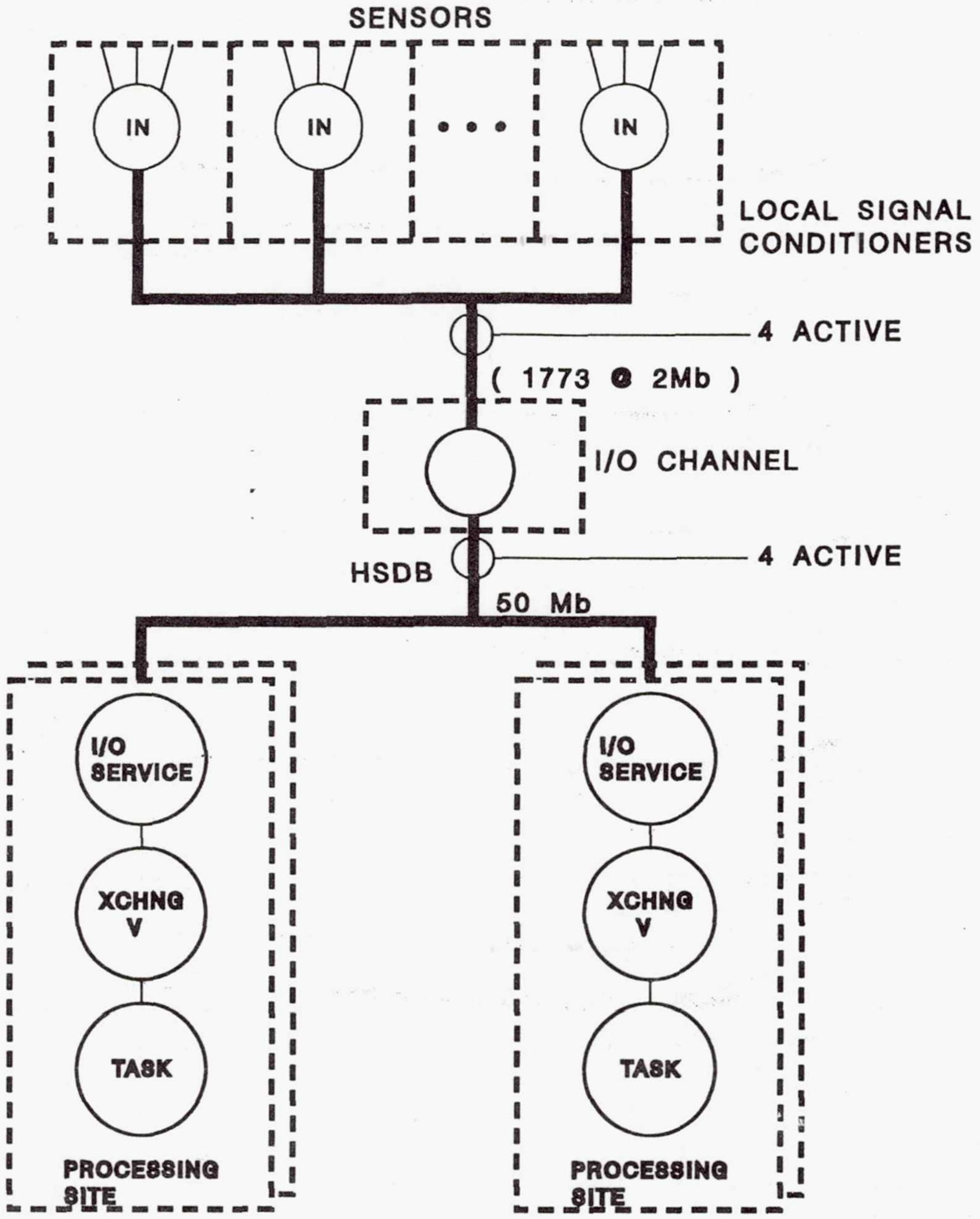


Figure 4.14. Boeing I/O Structure

1. the time required for data to be solicited from a local signal conditioner after an application task has requested data.
2. the time required for the local signal conditioner to respond.
3. the time required to check for errors in each step.
4. the time required to determine the destination of received data.
5. the time required for data formatting.
6. the time required to distribute data across redundant processing channels.
7. the time required to transfer data into and out of the various memories involved in the message delivery process.
8. the time required to reduce the redundant sensor data sets to a single consistent sensor data set for all operational channels.
9. the task switching times associated with each processor in the message delivery path.

None of the performance evaluations reported by Boeing indicate the extent to which the data delivery sequence, the associated error checking and the management of redundant sensor data were incorporated in the system performance evaluation. The data delivery time rather than bus bandwidth is the limiting factor for both I/O and interprocessor communications. It is more critical that this factor be analyzed for the Boeing architecture because of the more centralized processing topology proposed. That is, data from the large number of sensors are delivered to fewer core processing sites in the Boeing architecture and, consequently, a higher data bandwidth is required at each site.

Architecture elements such as the I/O channel module, the I/O interface module, and the system bus module must handle the data rate associated with the signal conditioners connected to the processing channel. Since redundant sensor data must be distributed to all processing channels, the fault tolerance module must handle the data rate associated with all local signal conditioners connected to the redundant channels of a processing site. While all delay factors in the data delivery mechanisms were of concern, the handling of sensor data in the fault tolerance module was examined more closely.

The functionality and use of the fault tolerance module is not described extensively in the Boeing specifications. With the exception of calling out a 3.5 MIP processor, the performance of the fault tolerance-model is not specified. Assuming that the module will be used to establish consistent sensor data across processors, to synchronize redundant channels, to vote partially processed results, to distribute error data and status across redundant channels, and to align processor states during recovery, only a portion of its capacity can safely be allocated to establishing consistent sensor data. A simple performance calculation can be made for the fault tolerance module. Assume that 25% of the fault tolerance module capacity can be dedicated to establishing sensor data consistency and that 500 kbps of the maximum specified sensor data rate, 1 mbps, must be handled by a processing site. Further, assume that the sensor data rate is uniformly distributed across quad redundant channels. The average time between 16-bit sensor data words in each channel is then 128 microseconds. A simple model of the relationship between the sensor data word interval and fault tolerance module performance is given by:

$$R(T_{DV} + T_S + I \cdot T_I) \leq C_{AS} T_{SD} \tag{4.1}$$

where:

R	=	redundancy
T_{DV}	=	time required to distribute and vote a simplex sensor data word
T_S	=	time skew between redundant processing channels
I	=	average number of instructions allocated to handling a word to be distributed and voted including memory transfers and error processing (These are instructions that cannot be overlapped with the distribution process)
T_I	=	average instruction time of the processor in the fault tolerance module
C_{AS}	=	% fault tolerance module capacity dedicated to sensor data consistency
T_{SD}	=	average interval between sensor data words within a processing channel

Note that all fault tolerance module times are multiplied by the redundancy factor R. This is due to the need to distribute copies of the redundant sensor data to all channels in order to assure consistent data in all operational channels. Note also that it is assumed that the skew between redundant channels impacts the voting of each word. The impact of channel skew depends on the design of the fault tolerance module which was not given by Boeing. This simple model accounts for establishing R sensor data sets in each processing channel. Additional processing would be required to establish a single voted sensor data set.

If a skew of 1 microsecond and 10 instructions are required for each word, the time to distribute and vote simplex data must be less than 4 microseconds per word or 4 mbps. This results in a reasonable specification for the fault tolerance module distribution/vote timing. If 20 instructions are required for each word, the time for distributing and voting must be less than 1 microsecond per word or 16 mbps. This would result in a much more demanding specification for the fault tolerance module. Recall that the AIPS requires in excess of 100 instructions on average for each sensor sample delivered to an application task. Thus, 10 instructions per word for this portion of the sensor delivery is very tightly coded. If the more demanding I/O rate of 6 Mbps as estimated by the Martin Marietta specifications must be met instead of the 1 Mbps rate, the fault tolerance module processor would have to be much faster than 3.5 MIPS, the channel skew would have to be reduced, the distributed vote time would have to be reduced and the 1773 busses would not be adequate.

Since the channel timing skew, the average number of instructions per word, the distribution/voting time and the module capacity dedicated to each fault tolerance module function such as synchronization or sensor data consistency, it is not possible to assess the adequacy of the Boeing design. It is further believed that the performance of the fault tolerance module is critical and, left unspecified and unanalyzed, represents a significant technical risk.

Since the proposed Boeing topology has only a GN&C processing site and a vehicle management processing site, most interfunction communication takes place within the processing sites and does not require the use of the system HSDB. Thus, the bandwidth of the bus and even the effective bandwidth of interprocessor communication software would not likely limit system communications. If Boeing's low estimates for propulsion control throughput do not hold and additional processing sites must be introduced into the Boeing design, substantial portions of interfunction I/O will be implemented via the system HSDB and the performance of the interprocessor communications software will have to be analyzed to determine its adequacy.

Further development of the Boeing MPRAS architecture should be predicated upon a better definition and performance analysis of the I/O and interprocessor communications software.

General Dynamics I/O and Intercomputer Communications – Figure 4.15 diagrams the hardware modules associated with a channel of a Remote Data Interface (RDI) or a processor in the General Dynamics architecture. An RDI channel is comprised of a sensor input module, a local data link module, a self-checking pair processor/memory module, a system bus module and an output module. The software that provides communications between application tasks and the sensors and actuators, and manages the network errors has not been specified for the General Dynamics architecture.

Illustrated in Figure 4.15 is a sequence of I/O data flow operations that is presumed for a Remote Data Interface (RDI) of the Boeing architecture. Sampled sensor data

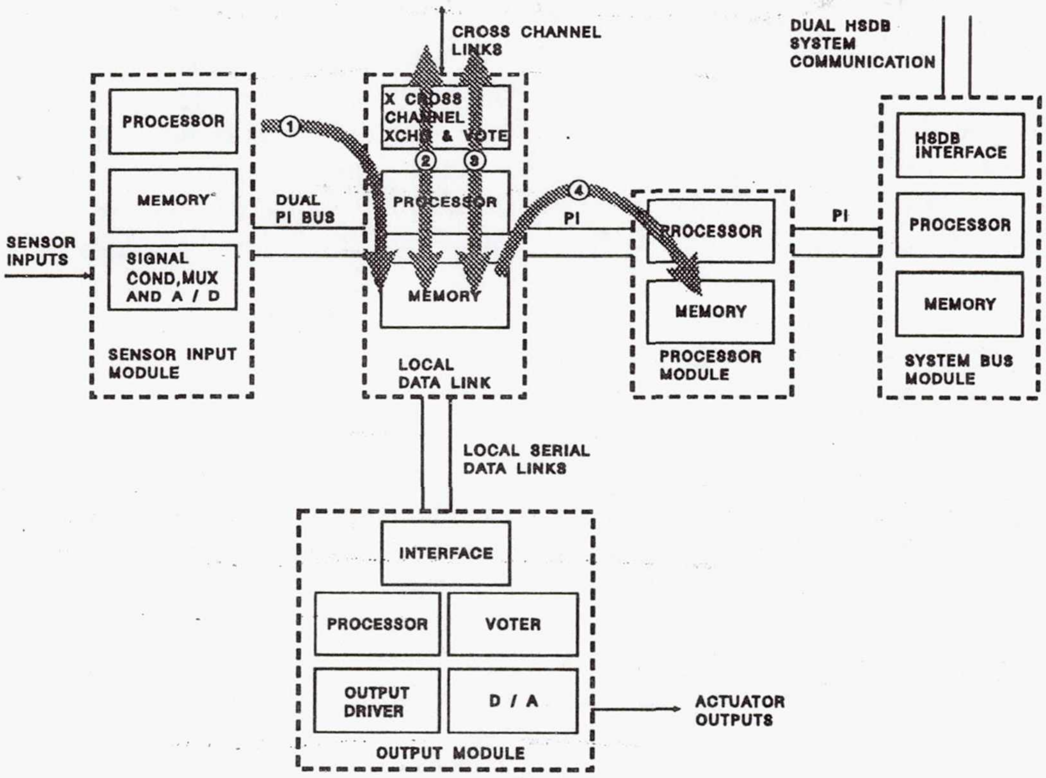


Figure 4.15. General Dynamics Sensor Data Flow

is transferred from the sensor input module to the local data link module. Sensor data is distributed to other redundant processing channels via the cross channel data links. Redundant sensor data is also received from other channels via the cross channel data links. Presumably, each channel will derive a single set of sensor data which will be assembled into a message and transferred to core processing sites such as the vehicle management processor via the redundant HSDB system data bus. Sensor data messages received by the core processing site will be processed for errors and directed to appropriate application tasks. Figure 4.16 shows the overall sensor data collection structure of the General Dynamics architecture. Multiple RDI's transfer redundant voted data to core processing sites via the HSDB.

Similar to the Boeing architecture, the software to handle the required sequence of operations to deliver consistent sensor data to application tasks has not be defined nor has its performance characteristics been analyzed and specified. Of concern is overall latency of data delivery and the effective sensor data communications bandwidth. The performance and functionality of the local data link module was not defined for MPRAS Part 1. It is assumed that this module will provide for distribution and voting to assure consistent sensor data in all channels, will provide for synchronization of redundant channels, will provide for alignment of the state of channels during recovery and will provide for distribution of error information and status between redundant channels. The distributed structure of the General Dynamics architecture can reduce the throughput requirements for the local data link module relative to that required for the Boeing fault tolerance module. By distributing the voting and data distribution function to local or regional data collection sites, the throughput requirements for any one site is reduced from that required for a more centralized approach. In this respect, a more distributed sensor data collection structure tends to be more expandable and places less demanding performance requirements on the local data link modules. However, unless properly designed, the additional voting planes required in the distributed architecture could result in higher costs.

As was the case with the Boeing fault tolerance module, the definition of the local data link module was not sufficient to assess the adequacy of its functionality and performance for the MPRAS application. Since this module is expected to provide the functions that are essential to the redundancy management of a multipath system, incomplete functional specification and performance analysis of this module represents a significant technical risk.

Communications between processing sites and the delivery of sensor data to processing sites is provided by the redundant HSDB system data bus. The General Dynamics computation model sets this communications rate at 12.8 mbps which is about 25% of the HSDB capacity. Capability to actually realize this

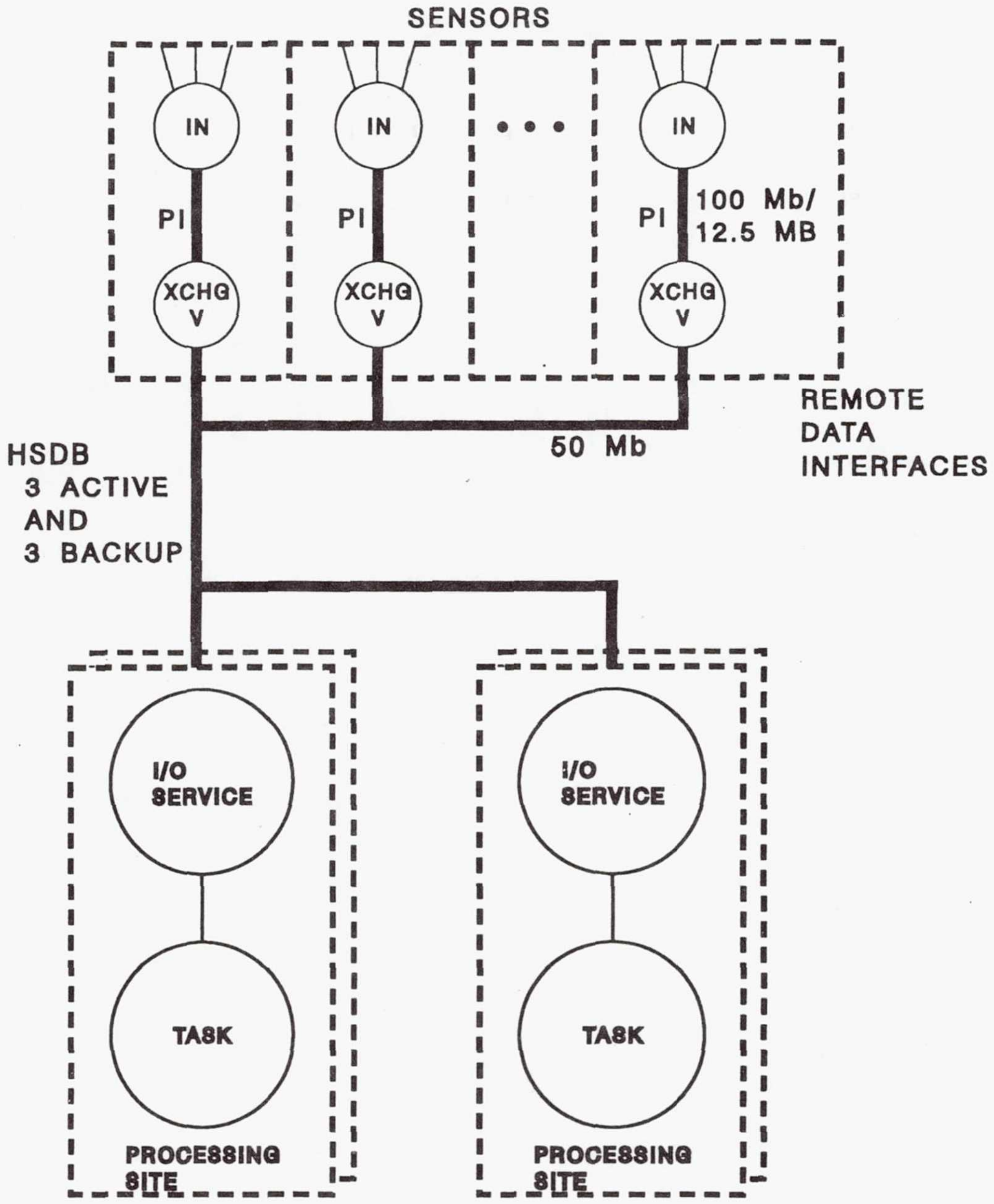


Figure 4.16. General Dynamics I/O Structure

communications bandwidth depends upon the software data delivery delays and the distribution of this requirement across the processing sites. Input requirements are highest for the vehicle management processor which provides control functions, the data recording group, and the telemetry processing group. The software that provides for preparation of data messages to be sent and for the delivery of data to an application task must be capable of sustaining the peak data rate while providing the necessary error detection and management capability. General Dynamics has proposed the Table Driven Proportional Access timing approach which was devised by Honeywell SRC. Using this approach, bus transmissions are prescheduled and message lengths and destinations are predetermined during the application software development process and occur at deterministic times. This approach has the potential to simplify and hence improve communications performance by eliminating the time required for contention on the system bus, by reducing the need for source, destination, and message length information in messages and by reducing the processing overhead required to deliver a message to an application task relative to that required by a general purpose network message handling software. Since this software has not been specified and described, it was not possible to analyze the performance of it.

To achieve the peak input rate to the vehicle management processor from the system bus of $0.9\text{x}10^6$ bytes/second, the HSDB interface module must complete its processing of a data word in an average of 2.2 microseconds. If the processor on the JIAWG HSDB module has the current throughput of 3.5 MIPS, approximately 7 instructions can be executed for each word handled. This would result in 100% loading of the HSDB module and would not allow for system output. Even without definition of the interprocessor communication software, it can be concluded that using the current standard JIAWG HSDB module is not feasible to meet the vehicle management processor input requirements as given in the General Dynamics MPRAS Point Design System Partitioning of August 2, 1989.

As with the Boeing architecture, further development of the General Dynamics MPRAS architecture should not proceed without more careful definition and performance analysis of the I/O and interprocessor communications software.

5. FAULT TOLERANCE FEATURES OF MPRAS ARCHITECTURES

5.1. Introduction

To provide the advanced functional features, such as adaptive guidance and control and vehicle health management, which are expected to enable substantial savings in launch operations, the complexity of the avionics hardware is much greater than that for current launch vehicles. A consequence of this increased complexity is an increase in unreliability. To offset this reduction in reliability and to further reduce launch vehicle costs due to unreliability, the use of fault-tolerant systems technology for the ALS avionics was examined under the MPRAS ADP's. While essential architecture characteristics such as performance and use of common modules had to be considered for MPRAS, the primary focus was to be on the infusion of fault-tolerant system technology into the ALS avionics. Identification of appropriate fault tolerance techniques, definition of suitable system development methods, assessment of potential benefits, limitations and technology risks and the preparation of technology development plans were among the goals of the MPRAS ADP's.

The designs of the fault tolerance features for the proposed MPRAS architectures were reviewed to determine adequacy, completeness and potential development risks. The primary areas focused on were: 1) fault masking and data consistency mechanisms, 2) fault detection mechanisms, 3) fault recovery techniques, 4) redundancy and sparing, and 5) validation and development methods.

5.2. Fault Masking and Data Consistency Mechanisms

5.2.1. Background

Fault masking provides fault tolerance through the use of redundancy to isolate or correct fault effects before erroneous outputs can propagate from a failed module. Each of the proposed MPRAS architectures relies on N-modular redundancy with voting to mask processing and computational errors. N-modular redundancy is based on the comparison of data from redundant channels to detect and mask faults. For this comparison to be effective, the following conditions are necessary:

1. The redundant processors have to be synchronized to a known and bounded timing skew.
2. The redundant processors must be initialized to consistent starting conditions.
3. All redundant processors must gather inputs and produce outputs in the same order.
4. Bitwise identical inputs must be used in the redundant processors.
5. A mechanism to detect bitwise disagreement between redundant processors must be used.

Synchronization is needed to establish an *a priori* limit on processing time so that a slow non-failed processor can be differentiated from an inactive failed processor, the comparator can know when valid outputs have been received, and duplicated sites can achieve bitwise consensus on input data. It can be achieved by communication between channels, which is subject to faults in either channel or in the inter-channel communication mechanisms, or by the provision of a reliable clocking signal on each channel to which all channels synchronize.

The requirement for bitwise identical computation and output agreement requires bitwise identical input, which in turn requires that all channels use identical or consistent input data. To assure that identical inputs are distributed to all channels in the presence of arbitrary failures, an input consistency protocol is required that is f-Byzantine resilient; i.e., can tolerate f arbitrary failures. For an input consistency protocol to be f-Byzantine resilient requires: 1) at least 3f+1 participants, 2) each participant to be connected to at least 2f+1 other participants through disjoint communication paths, 3) at least f+1 rounds of communication among the participants, and 4) the synchronization of the participants within a known skew.

The participants in the protocol need to be fault containment regions; i.e., regions to which faults can be sufficiently contained to ensure the statistical independence of failures in any two regions. Otherwise, single failures could result in the simultaneous loss of more than one of the redundant channels. Therefore, four FCR's are required to tolerate one arbitrary failure. Four architectural features lead to independence of failure: 1) physical isolation, 2) electrical isolation, 3) independent power, and 4) independent clocking.

Fault masking and data consistency mechanisms are essential elements for achieving fault tolerance with a multipath redundant system. Unless these mechanisms are properly designed, the potential reliability of a multipath system cannot be realized. So important are these mechanisms that an architecture design cannot credibly be represented as fault tolerant unless these mechanisms are rather fully specified as to functionality and performance. Proceeding with an architecture development without such specifications should be considered a technical and development risk. The mechanisms which provide fault masking and data consistency for each proposed MPRAS architecture are discussed in the following paragraphs.

5.2.2. General Dynamics Fault Masking

Output Voting – Digital outputs from each redundant string are voted bit-for-bit in the General Dynamics output modules. Each redundant string may have an output module capable of driving multiple actuators. The voted outputs from each string are converted and then used to drive redundant actuators. Up to four redundant channels can be voted. The specification of the output module functionality and performance is incomplete in several essential areas. Among those characteristics left unspecified are: 1) the nominal timing skew allowed between redundant strings and the output data/command buffering required to align skewed outputs at voter inputs, 2) how an output module responds if a redundant string fails to deliver data to be voted, 3) if and how a bad string can be excluded from a vote, 4) if and how minority strings are identified and reported, 5) if and how the voter and minority reporting are checked during operation, and 6) the data bandwidth of the voter and drive converter elements. Without specification of these essential functions, the design of the output fault masking is incomplete and its adequacy cannot be assessed. No specific redundant channel fault masking is provided for communications on the system bus. Accordingly, the failure rates for all processing sites within a channel along with the failure rates of the bus communication paths are added to obtain the failure rate for a single string.

Input Data Consistency – Voting of output data values depends upon each redundant channel having consistent input data. The General Dynamics architecture provides for a Local Data Link (LDL) module which can be used to establish consistent data in each redundant channel. Cross channel communication paths are used to pass data between LDL modules in redundant channels. Redundant data from multiple strings can be voted in the LDL. Communications paths are optically isolated to maintain independent fault containment areas. This prevents failures on a link driver or receiver from causing all LDL's to fail. It is implied that the communication paths will be either self-checking paths or will use error detection and correction coding to ensure reliable cross channel communications. A description of the options for establishing consistent input is given in the following paragraphs.

The architecture uses two methods to ensure consistency of the sensor input data-syntactic error checks and semantic checks. When possible, the architecture depends on syntactic error checks to detect potential faults. Where syntactic checks are not effective, it relies on semantic checks to detect errors based on the value or reasonableness of the data. Semantic checks are dependent on application and therefore left up to the programmer to implement. Syntactic checks, such as a bit-for-bit vote of the three redundant channel outputs, are performed automatically by the system. The LDL provides the hardware means by which these syntactic checks are performed. The RDI's, designed to collect sensor data and develop a single, consistent input value for each channel, can be configured in a variety of options depending on the criticality of the sensor data and the redundancy level of the sensor. In the case of a single flight critical sensor, two options are specified. The first option ties the sensor to channel A of the RDI and the data value is distributed to channels B and C via the LDL data exchange. This option does not guarantee input consistency in the presence of arbitrary failures in channel A.

The second option cross-straps the sensor to four channels: A,B, and C and a fourth consisting of a sensor interface and an LDL module. The addition of the fourth channel provides the fourth fault containment area which is the basis for a Byzantine-resilient protocol. However, it is not clear whether the required two rounds of communication are to be provided. The guidance given for using the sensor cross strapping option instead of the capture then distribute option was that the reliability of the sensor and the cross strap wiring should be much higher than that of the sensor interface and the LDL module. It is RTI's opinion that the decision must also include the probability that for certain upset events such as lightning strikes, the sensor cross strap wiring makes all channels vulnerable to damage. That is, use of sensor cross strapping can make the system more susceptible to non-recoverable common mode failures.

Instead of using the above approach to distributing consistent input data, an approach which provides the appropriate number of fault containment regions and disjoint communication paths for a protocol that tolerates arbitrary behavior could have been considered for the LDL. When error correcting or self checking communication paths, multiple rounds of communications and the extra sensor input and LDL module proposed by General Dynamics are taken into consideration, this approach could in fact be less complex and expensive. Important unspecified characteristics of the LDL function include: 1) the maximum timing skew between redundant channels, 2) the impact of the skew upon LDL throughput, 3) the buffering required to align voter inputs in the presence of skew, 4) the behavior of the LDL when the LDL in a redundant string fails to deliver expected data, 5) if and how a faulty channel is excluded from a vote, 6) if and how a minority string is reported, 7) if and how the voter and error report logic are checked, 8) the necessary bandwidth for the LDL function, and 9) the specifics of the cross channel link self-checking or error correction functions. Without specification of the essential functions, the design of the LDL is incomplete and its adequacy cannot be assessed. As such, this represents a significant development risk.

Synchronization – There are three types of synchronization that must be carried out for the General Dynamics architecture. The first type is the synchronization of the two processors in the self-checking pair (SCP) processor module. The second type is the establishment of synchronization of all processing elements within a redundant channel. The third type is the synchronization of the redundant strings.

The synchronization of processing elements within a channel provides the basis for communication on the system bus. Communication between Bus Interface Units (BIU's) within the architecture is accomplished with decentralized bus control via Table Driven Proportional Access (TDPA). TDPA is a time-multiplexed scheme based on a coordinated allocation vector in each BIU. This permits processors to share a bus without an explicit bus controller. Connections to the bus consist of both transmit/receive and receive only ports. Each BIU has a unique (for its transmitting bus) identification (ID) code which determines when a particular module is allowed to transmit to the system bus. Each processor contains a bus access table or vector. The access table allows only one module access to the bus at any one time. Using the same access table, each module's BIU checks to see if its ID number matches the access table's current time slot ID as listed in the access table. If so, that module has transmission rights on the system bus at that time. Proper maintenance of the access table pointer by all system modules maintains synchronization across the system. Each processor has special algorithms that maintain synchronization with the system bus and provide for resynchronization to the bus when required. The length of the access table and the bus clock rate impose a frame rate for data transfer on a given bus. Because the data type to be transmitted in a particular slot is known ahead of time, it is possible to allocate a tight message window time to each slot. The window is large enough to accommodate a message of the prescribed number of words along with tolerances for timing skews between processors. Each TDPA frame is composed of a number of time slots. Each time slot has a message window and an inter-message gap. The message window is made large enough to accommodate a message of the indicated data type as well as tolerances on either side for BIU-to-BIU timing skew. The inter-message gap is a fixed size determined largely by the amount of time required by each BIU to set up for the next message. The timing skew between BIU's and the synchronization and resynchronization algorithms have not been specified. A non-responding BIU does not affect the timing of any other messages sent by other modules. Therefore, the time to run completely through the table is always known and is independent of the number of timed-out messages as long as there is at least one message transmitted during each cycle through the table. This mechanism is said to yield completely time-determinate system operation and that the access table can be configured to insure that time critical data transfers are supported.

Establishing and maintaining synchronization of redundant strings is necessary to satisfy the bounded skew requirements for voting of redundant processor outputs and for establishing consistent input data. The LDL modules in a string provide the only cross channel communication paths by which cross channel synchronization can be achieved. Description of this process for the General Dynamics architecture is limited to the following statements: 1) "The exchange of data across channels tends to provide inter-channel synchronization. This exchange of information provides a means by which the skew between channels is limited and directly affects any time-out functions implemented within the system." and 2) "Strings synchronize with each other at message boundaries according to the TDPA mechanism." This description is insufficient to determine the functionality and adequacy of redundant channel synchronization. This description leaves the actual bound on time skew unspecified. Note that in addition to affecting time-out functions, skew between redundant channels can add to the sample data control loop latency or lag, can limit the coherency of sensor data samples and, depending upon LDL design details, may limit LDL bandwidth. Note also that the skew within a string may be additive with the cross channel skew in turn producing a larger overall skew. Also left unspecified is the explicit mechanism or algorithm by which synchronization is established initially, maintained in the face of relative timing drift and re-established following transient faults. Since there are multiple processing elements and associated LDL modules within the system, it is not clear if one set or all sets are involved in this cross channel synchronization process. If more than one set is involved, their mutual coordination may need to be specified. If only one set is involved, the designation of this master set and the potential need for alternate masters may need to be specified.

Without more complete specification of the synchronization of redundant channels, the fault tolerance of the architecture cannot be evaluated. The role of this synchronization is of such importance to the fault tolerance of a multipath system that development of the architecture should not proceed without a more complete specification of this function.

Error Correction Features – Error correction techniques can be employed to mask errors in data transmission and storage. Inherent in the use of JIAWG modules for the General Dynamics architecture is the use of a Hamming code to correct or mask transmission errors on the PI bus. This is the only specific use of error correction techniques that can be identified for this architecture. Other areas where the use of error correction has been suggested but not specified include the use of non-specific error correction coding for the LDL cross-channel data links, error correction for memory bit errors and allocation of spare transmission windows in the TDPA template for retransmission of messages that previously contained errors. Presumably, software would have to be included to respond to message errors with an appropriate retry. Consideration should be given to the use of memory error correction in the self-checking processor/memory module. The purpose would be to reduce the frequency at which the self-checking logic pulls the SCP off-line due to soft errors in the memories.

5.2.3. Boeing Fault Masking

Input Data Consistency and Output Voting – The fault tolerance module (FTM) of the Boeing architecture implements the voting and synchronization functions. It will be composed of a 1750A processor, a 256K memory, a maintenance controller and interface, a PI bus interface, a switched or voted redundant system bus interface and an inter-channel communication interface with voter and synchronization logic. Boeing also indicates that rigorous fault tolerance concepts will be used. Boeing defines the degree of electrical isolation that is expected to constitute a physical fault containment region. This is the extent of the functionality and performance specifications for the FTM. As with the General Dynamics architecture, the specification of essential features is incomplete and the potential for the architecture to achieve expected fault tolerance cannot be evaluated. Among the essential characteristics which should be specified and evaluated before development proceeds are: 1) the nominal skew expected between redundant channels, 2) the fault containment characteristics of the FTM, 3) the behavior of the FTM when a channel fails to deliver data to be voted, 4) if and how faulty channels are excluded from voting, 5) if and how a string whose data is in minority is reported, 6) if and how the voter and error report logic are checked for faults during operation, 7) the mechanism for distributing consistent simplex data, 8) the connectivity of the inter-channel communication paths, 9) the bandwidth of the inter-channel communication paths and associated voting and error reporting functions, and 10) if and how fault masking is provided for system bus communications between processing sites. The design of the switched or voted system bus interface is not indicated. While the interface can provide masking of errors in intercomputer communications, unless designed properly it has the potential to permit malicious behavior in one processing channel to disrupt the other redundant channels.

Synchronization – All redundant channels within a processor will be powered-up together at the start of power-up. Following self-test of all of the channels, the individual channels will be synchronized to each other via the FTM hardware and self-test will then be conducted at the node level. Task or frame level synchronization will be used. A real-time clock in the Guidance, Navigation and Control (GN&C) processor will be used as the reference to which all elements synchronize. A spare processor will be capable of taking over these functions if the GN&C processor fails. Real-time clock synchronization will control drift.

Without more complete specification of the synchronization process, the capability to establish, maintain and re-establish, if necessary, a bounded timing skew across redundant channels cannot be evaluated. Further, the fault tolerance of this function in the presence of arbitrary failures cannot be assessed.

Error Correction Features – Specific error correction techniques other than redundant channel voting identified for the Boeing architecture include the error correction coding inherent in the JIAWG PI bus interfaces and the error correction coding present in the JIAWG bulk memory modules. Retry of transmissions for messages with errors is specified for data busses and backplanes.

5.2.4. AIPS Fault Masking

Data consistency and output voting is provided in the AIPS FTP via the communicator/interstage hardware. This hardware has the connectivity and physical fault containment regions necessary to provide for distribution of consistent data in the presence of arbitrary failures. It is one-fault Byzantine resilient in a triplex configuration and two-fault resilient in a quadruplex configuration. The hardware provides for bit-for-bit majority voting, reporting of errors in the voting process and the capability to block faulty processing channels from the vote process. The function has been implemented and tested extensively in a series of implementations.

Error reporting circuits and voter logic are tested exhaustively by a diagnostic test program executing on a time available basis in the FTP. Data paths and control circuit logic are tested through normal use of the hardware and through a special test executed at the beginning of each processing frame.

The minimum time required to vote a word is 2.5 microseconds in the proof-of-concept model of the FTP. The fault-tolerant clock used to cycle the voter has a 5 microsecond period and is controlled by redundant digital control loops which can adjust the fault-tolerant clock skews by 125 nanoseconds every fault-tolerant clock period. The technology used for MPRAS should permit these parameters to be reduced.

Upon start-up and for resynchronization, operational redundant FTP channels are synchronized to an instruction interval and to identical hardware states by a software service. Initial synchronization is attained by exchanging unique words via the communicator/interstage hardware.

Communications between FTP's is provided via a three-layer IC network. FTP channels can receive on all three network layers either by voting data received on all layers or by selecting data from a single layer. FTP channels can transmit on only one layer of the network. Transmission errors for data transmitted on all layers are masked by a voter in the FTP IC interface. Data transmitted on only one layer are received by all channels in the receiving FTP. A consistent copy is formed by voting the received copies via the communicator/interstage hardware. Error correction of redundant memory values via a memory scrub program that executes as a background self-test corrects for soft memory errors and prevents the accumulation of these errors over long periods of operation. This mechanism makes use of the voter hardware to find and isolate these errors.

5.3. Fault Detection and Diagnosis

Fault detection relies on the provision and use of redundant information or resources to detect the faults and errors caused by failures. The N-modular redundancy with voting strategies that are being used in the proposed MPRAS architectures provide the basis for fault detection, provided that they are augmented with appropriate and effective fault diagnostics. The major issues are diagnosability and coverage. Also, as for fault masking, for N-modular redundancy with voting to be effective, the following conditions are necessary:

1. The redundant processors have to be synchronized to a known and bounded skew.
2. Bitwise identical inputs must be provided to the redundant processors.
3. A mechanism to detect bitwise disagreement between redundant processors must be used.

Although the fault masking provided by majority voting is sufficient to provide fault tolerance, additional diagnostic and fault detection capabilities may be required to support redundancy management requirements, including notification of fault occurrence and location of the fault to some finite number of possible failure locations. These two functions are essential if timely reconfiguration is needed to maintain the level of redundancy after the occurrence of failures or to remove faulty components.

Each of the proposed MPRAS architectures provide error detection and identification capabilities beyond that provided via the majority voting mechanisms. Those capabilities excluding the built-in-test features which support thorough assembly and prelaunch readiness testing are discussed in the following paragraphs.

General Dynamics requirements call for the use of "state-of-the-art hardware and software techniques to establish a high level of system tolerance to both hardware and software errors." "The ability to detect, correct or compensate for soft errors induced by transient hardware or environmental anomalies is required."

Specific error detection capabilities that are inherent in the use of JIAWG common modules and would be applicable to the General Dynamics architecture are: 1) error detection and correction in bulk memories, 2) memory parity checking, and 3) error detection and correction on the PI bus interface.

The primary error detection method put forth for the system bus is the use of an extra bus for each redundant bus channel along with self-checking bus interface modules. This would require a bus interface module that differs from the JIAWG standard interface module. If this approach is not used, the extra busses would be used as spares and unspecified error checking words would be included and tested for each message transmitted.

The primary error checking mechanism proposed for the General Dynamics architecture is a processor/memory module which uses the self-checking pair concept to achieve high coverage and rapid detection of errors. Self-checking pairs (SCP's) are used in all redundant channels to provide rapid fault detection, isolation and containment at the module level. The use of SCP's simplify the diagnostics that are necessary to perform these functions and thereby speed up the recovery process so that spare modules can be used to maintain the desired redundancy levels after multiple failures. However, the SCP's still need to execute diagnostic sequences to check the comparator and to determine whether faults are transient or permanent.

Each SCP consists of 2 lock-step processors, a fault monitor module and a PI bus interface. A self-checking comparator is part of the fault monitor. The self-checking monitor checks itself while it checks the processor outputs. It also checks that processor clocks are correct and indicated an error if they are not. A self-checking fault management sequencer in the fault monitor module disconnects the SCP from the system bus when an error is detected, diagnoses transient faults and allows the SCP to re-establish normal system operation when the fault is diagnosed to be transient. The processors that make up the SCP are run in lock-step.

The self-checking fault management sequencer in the fault monitor module of the SCP is a simple device comprising counters and combinational logic. If processor disagreement occurs or the self-checking comparator finds an internal fault, the sequencer transitions from the "good" state to the "abort" state. In the abort state, it signals an SCP fault and resets the processors. The external fault signal is transmitted over the Test and Maintenance Bus and PI bus to ensure that errors affecting the transmission or generation logic are also detected. Then the sequencer unconditionally transitions to the "disable PI bus" state, where SCP outputs to the PI bus are disabled. PI bus outputs were left enabled in "abort" state to allow the sequencer to signal the malfunction. Processor resets are released in the disable PI bus state to allow them to execute a diagnostic sequence. The diagnostic sequence exercises all of the SCP module hardware including the processors and local memory. The processors send a command to the sequencer when they have successfully completed the diagnostic sequence. The sequencer responds by transitioning to "re-enable" state to allow the SCP module to communicate over the PI bus. The sequencer then transitions back to the "good" state which allows the SCP module to be reused as a spare. The recovery process requires the processor to obtain software state and database information from one of the active processors. This would normally be performed after the module is brought online as an active module. The sequencer transitions to the "crowbar" state which permanently disables the SCP module if during the "disable PI bus" state 1) the processors disagree, 2) the check circuit has an internal fault, 3) there is a sequencer fault during the diagnostic period, or 4) the processors fail to complete the diagnostics within a specified period of time. The sequencer can permanently disable the module either by continuously maintaining the reset signal true, or by causing power to be removed from the module.

The use of SCP's has the following potential advantages:

1. Processors do not need to send "keep alive" signals to the fault monitor logic.
2. Eliminates need for detailed fault analysis for the processors since all non common cause faults will result in processor disagreements.
3. Immediate disabling of outputs eliminates possibility that errors will propagate.
4. Reuse of SCP if fault is determined to be transient.
5. Potential higher coverage and reduced latency for faults.

The use of SCP's has the following liabilities:

1. Comparator is a single-point failure; careful design is required for reliability.
2. They are potentially more prone to common mode failures.
3. The fact that the checker exists in the same fault containment region as the devices being checked makes it difficult to guarantee independence of failure modes.
4. SCP failure rates are at least double single processor failure rates.

Specific error detection capabilities that are inherent in the use of JIAWG common modules and which are applicable to the Boeing architecture are: 1) error detection and correction in bulk memories, 2) memory parity checking, and 3) error detection and correction on the PI bus interface. In addition, Boeing indicates the inclusion of error checking words for all message transmissions on various system busses. The Boeing specifications present a relatively thorough analysis of a broad range of faults within their architecture and indicate how each fault is covered by the various error detection mechanisms. Fault and error assessment processing will be implemented in the Boeing FTM. The Boeing specifications call for each processing site to carry out a self test on power up to establish its readiness to begin processing.

Fault identification algorithms for both the General Dynamics and Boeing architectures cannot be specified until their designs progress further. However, certain important characteristics such as the methods for assuring consistent error report inputs to the fault diagnosis process and for assuring that the non faulty channels arrive at the same diagnosis should be, but have not been, specified.

Extensive error detection mechanisms have been implemented for the AIPS proof-of-concept system. Each processor detects the following exception errors: 1) bus errors, 2) address errors, 3) illegal instruction errors, 4) arithmetic traps, and 5) spurious interrupts. A watch dog timer is used to detect processors that fail to complete operation sequences. The intercomputer and input/output networks detect protocol errors, data errors and time-outs. In addition to the detection of data errors on network communication paths, messages contain extra error check words that detect data errors in messages after message data has been handled within FTP memories. Voters are used to mask errors on intercomputer communications. On-line diagnostic self-test programs are executed during available processing intervals to detect faults in the voter logic, error reporting circuits, data memory, program memory, the real-time clock and the monitor interlock. Presence tests are executed every processing frame prior to application processing to establish which channels of each FTP are available. In addition to these error detection mechanisms, fault diagnosis algorithms have been implemented to analyze these error reports along with errors detected by the voters to identify faulty elements. Note that proper fault diagnosis for a redundant system requires that all operational channels arrive at the same diagnosis. Consequently, the AIPS uses the data exchange mechanism to distribute to all channels the error reports from each channel so that consistent error report inputs can be used for diagnosis in each channel. Testing has been conducted to determine the correctness and effectiveness of these algorithms. Test programs which exercise the spare links and nodes in the intercomputer and I/O networks are executed to detect failure in these spare components. FTP's also execute a full self-test sequence following power up.

5.4. Fault Recovery

Recovery from faults in the AIPS architecture will be triggered by the fault detection and identification mechanisms discussed in the previous section. Initial response to all diagnosed channel faults is to block the faulty channel from voting. Following this, attempts are made to recover this channel. A channel is considered recovered if it can be resynchronized and state aligned to the operational channels and if there are no data exchange voting errors being produced in the recovering channel after a suitable period following resynchronization and state alignment. Faults detected and identified for the AIPS I/O networks and IC network will cause the networks to be reconfigured around the faulty component using spare communication links and nodes.

Since network reconfiguration and channel recovery require substantial time to complete, the AIPS architecture recommended for MPRAS will only make use of these fault recovery features during the pre-launch period. During the launch phase only fault masking and the capability to block channels diagnosed as faulty from the voting process will be in force.

Boeing has not specified or described the error diagnosis, reconfiguration, transient recovery, resynchronization and state alignment processes which are essential for fault recovery. Critical aspects of these processes need to be specified before their adequacy can be assessed and before the design can be considered to represent a credible fault-tolerant system.

The basic recovery process for the General Dynamics SCP was described in the previous section. In this architecture, recovery occurs at the subsystem level and the goal is for rapid in-flight recover. The capability to properly and rapidly introduce a spare SCP into the channel or to recover the faulty SCP from a transient depends upon self-test, synchronization, state recovery and state alignment processes. These processes must be specified and analyzed in more detail before their feasibility and adequacy can be established. Of particular concern should be the time required to align the state of the SCP memories with that of the other channels. This time could well exceed the specified recovery time.

The rapid in-flight recovery requirement specified by General Dynamics and Boeing is a major contrast to the in-flight fault masking approach taken for the AIPS. Since this requirement can have a substantial effect on the feasibility, cost and development risk associated with the MPRAS architectures and could be a discriminate between architectures, the need for this requirement must be resolved.

5.5. Redundancy and Sparing

The redundancy of the General Dynamics architecture can support up to quadruplex configurations. Triplex has been recommended as the baseline configuration. Due to the "two-failure" requirement for safety, launch will not be permitted if including the failure of spares the system degrades below the triplex level during pre-launch. The recommended redundancy for both the Boeing and AIPS architectures is quadruplex. These systems have to degrade from quadruplex to duplex during pre-launch before launch would be aborted.

Sparing in the General Dynamics architecture includes a spare PI bus for each backplane in a redundant channel, a spare system bus for each redundant channel, provisions for spare processors in redundant channels and the capability to provide spare processing sites at the system level. Spare processing sites can be used to replace any processing sites that are not used as an RDI; that is, is not connected to sensors or actuators.

Sparing for the Boeing architecture includes a spare PI bus for each backplane in a redundant channel and pooled system level spare processing sites. Due to the Boeing core processing and local signal conditioner topology which permits all processing sites to have access to all sensors and actuators, the spare processing site can be used to replace any processing site. AIPS permits sparing at the fault masking group or processing site level. The capability of the spare to replace a fault masking group depends upon whether it can be given access to the I/O devices (sensors and actuators) connected to the failed site. Access to an I/O network requires an IOS module. Thus, sparing will be limited to the number of distinct I/O networks for which distinct IOS modules are provided in the spare. If a system has a number of

I/O networks, sparing could be constrained. AIPS building blocks also provide the opportunity to include spare communication links and circuit switched network nodes within the I/O and IC networks. These spare components can be used to reconfigure these network components to bypass faulty network components.

6. SUMMARY AND CONCLUSIONS

The following paragraphs summarize the conclusions that have been drawn regarding the MPRAS architectures.

Reliability – Reliability projections were made by Boeing for recoverable core avionics over multiple missions. In addition, Draper Laboratory used mid 80's failure rates to evaluate the reliability of an AIPS core avionics for a mission consisting of a pre-launch and launch phases. No system reliability projections were made by General Dynamics for their architecture. None of the contractors conducted reliability analyses of the sensor/actuator communications networks. Projections indicate that this area has a substantial impact on MPRAS reliability and could cause overall MPRAS reliability to fall short of objectives. In particular, the triple redundant General Dynamics sensor acquisition network topology could fall far short of desired goals. The reliability of the sensor/actuator communications hardware should be considered a risk area for all MPRAS architectures.

The major factor impacting reliability projections for MPRAS is failure rate assumed for transient failures under both launch and pre-launch conditions. Establishing credible transient failure rates should be a priority item for MPRAS development. Otherwise, the system could suffer the costs of overdesign or suffer the costs of failure to fulfill mission requirements.

No reliability analyses of the built-in-test features were reported for the MPRAS architectures. The substantial built-in-test features required to enable cost savings in vehicle assembly, integration, and launch operations require sufficient hardware that the reliability of this hardware could adversely affect MPRAS reliability. The design of the testability features should be reviewed with the goal of improving reliability of the overall test system. The effectiveness of the entire error detection and isolation features of the MPRAS architectures has been shown to be a critical parameter for mission success and as such should be an area for scrutiny throughout the MPRAS development.

Testability – Both the General Dynamics and Boeing testability design specifications are at a level consistent with the development stage and are appropriate for the application. Both designs provide for off-line chip level built-in-test circuitry appropriate for reducing the otherwise enormous testing task presented by complex systems implemented with VLSI technology. The Boeing development is more advanced in specifying the testability features for the sensor/actuator signal conditioning/interface elements. The Boeing design provides for a test and maintenance bus within an avionics enclosure and provides for an external test and maintenance connection port on each avionics enclosure but does not carry a dedicated test and maintenance bus to all enclosures throughout the vehicle. The General Dynamics design is based on a dedicated test and maintenance bus that is provided to each avionics processing enclosure. The bus can provide access to points within the system that should result in better system testability. Information regarding the testability of the AIPS architecture for the projected MPRAS application was not available. However, the lack of chip level built-in-test capability appropriate for VLSI complexity in the AIPS proof-of-concept system is a significant disadvantage for assembly and pre-launch testing for MPRAS.

Because of the importance of testability to enabling costs savings in launch vehicle assembly, test, and operations; competitive MPRAS architectures should be carefully scrutinized with respect to adequacy and cost effectiveness. Due to the large amount of hardware required for an MPRAS application, the reliability of the associated built-in-test and maintenance hardware is a potential risk area that must be addressed for all architectures.

Performance – Because there are substantial differences in projected performance requirements for certain functions given in the three application descriptions, conclusions regarding the adequacy of architecture performance characteristics are subject to errors. Improved application descriptions and performance requirements should be developed before substantial MPRAS development is undertaken. Areas of most concern are propulsion control and adaptive GN&C computation and communication resource requirements.

Intercomputer and sensor/actuator communications are judged to represent the greatest performance risk for each architecture. The overall effective communication bandwidth which includes the effect of software processing for high coverage error detection and message delivery, as well as the cable bandwidth, is the relevant performance parameter of interest. The potential for efficient, low-overhead communications and task scheduling provided by the TDPA concept and the distributed sensor data voting topology are considered performance advantages for the General Dynamics architecture. Even though the AIPS communications mechanisms are software intensive, the fact that they have been implemented, that their message delivery times are known, and that the required improvement of a factor of 10 to 20 is feasible using projected technology, reduces the risk for AIPS performance. Due to the centralized topology of Boeing architecture, including the more centralized sensor data voting characteristics, it is the architecture that would be most affected by the need for higher throughput requirements such as those put forth for the propulsion control functions. Further, the topology seems to constrain the location of time-critical sensors to the PA module. This would preclude sensors for fuel level and structures which must be distributed throughout the vehicle and which are expected to be used for adaptive GN&C functions from being handled by the time-critical I/O mechanisms. Thus, the Boeing architecture development would carry a high performance risk.

Fault Tolerance – As indicated in Chapter 5, the fault tolerance features of the Boeing and General Dynamics architectures are insufficiently specified or reported to determine their adequacy or feasibility. For each architecture, numerous critical fault detection, fault diagnosis, fault recovery, fault masking and redundancy management characteristics must be defined before adequacy, feasibility, and development risk can be determined or before these designs can be represented as credible candidates for a fault-tolerant architecture.

The AIPS fault tolerance features are well-established and have been implemented in a series of development systems. These design features have undergone extensive analysis and testing. The verification and validation of the AIPS FTP and its associated local system service software has received the most attention. This process continues. Due to the advanced state of development, the AIPS fault tolerance concepts have by far the lowest technical and development risks for MPRAS.

The General Dynamics proposed requirement for rapid in-flight recovery can have a major impact on the MPRAS fault tolerance requirements. If this proposed requirement is in fact necessary, all architectures including AIPS will be significantly impacted. If, however, in-flight fault masking is all that is required, the proposed non-reconfiguring AIPS architecture can be used.

Susceptibility of multipath systems to the common mode (non-independent) failure is an obstacle for the acceptance of such systems for applications such as launch vehicle avionics. Additional work must be done to determine if this threat is significant and to devise ways to reduce this threat or its effect on systems.

Both Boeing and General Dynamics indicate that the SDIO BM/C^3 working group reliable system design framework will be used to develop their architectures. If this framework is augmented by appropriate methods and carried out carefully, it should be sufficient to reduce development and technical risks.

The AIPS application development concept is built on having validated "building blocks" and appropriate application guidelines such that configuring an application from them will require that only the application be validated. Substantial effort has been directed toward this goal and validation is relatively advanced. However, total architecture validation has not been accomplished nor has the concept of separating architecture and application validation been demonstrated. AIPS will have development support tools and an extensive performance information base which can be used to reduce development risks.

The General Dynamics TDPA concept introduces deterministic application timing which should greatly reduce the application validation or revalidation requirements and costs.

Common Modules and Adherence to Standards – Both General Dynamics and Boeing Aerospace have set requirements to adhere to or meet a comprehensive list of standards for materials, workmanship, processes, packaging, practices, environment factors and maintainability. JIAWG standard line replaceable module sizes and standard connectors are specified for packaging. Standard interfaces such as PI Bus, TM Bus, MIL-STD-1773 and the High Speed Data Bus are used. Common JIAWG modules or upgraded JIAWG modules are being used to the extent possible. The common avionics processor CAP-32 and the MIL-STD-1750 instruction architecture are also specified. MIL-STD-1815 Ada programming language is naturally dictated.

Based on the adherence to these standards and the maximum possible use of JIAWG common modules, it is indicated that costs will be reduced. Acquisition costs are reduced because high production common modules will cost less. Logistics costs will be reduced because of the adherence to standards. JIAWG module acquisition costs have been estimated at between $10 thousand and $15 thousand for typical modules. The costs for upgraded or new modules have not been formally reported but informal speculation sets these costs as high as $20 thousand to $40 thousand. No life-cycle cost reports addressing the avionics were available for the MPRAS Part 1.

Boeing Aerospace specifications called for the use of JIAWG modules. General Dynamics on the other hand indicated that the JIAWG modules would have to be upgraded to meet launch vehicle specifications. The primary question is how much of the cost advantage for common modules is lost due to upgraded repackaging. That is, when is a common module no longer a common module? Both Boeing and General Dynamics specify new modules such as sensor/actuator interfaces, fault tolerance modules, self-checking processor module and high throughput processors. These modules would not necessarily be used in programs other than ALS. There are about as many unique module types as there are common module types for both architectures and the total number of unique modules used in applications seems to equal or dominate the total number of common modules used.

A significant difference in the Boeing Aerospace design and the General Dynamics design is the use of liquid cooling. General Dynamics proposes to keep module power dissipation to between 10 and 15 watts so that the cost of liquid cooling can be avoided. Since current JIAWG modules such as a 4 MIP processor requires 35-40 watts and since 10 MIP self-checking pair processor modules are proposed, the feasibility of not having liquid cooling must be demonstrated.

The General Dynamics sensor/actuator conditioning and interface modules for

MPRAS are more completely specified, while the Boeing sensor/actuator conditioning and interface module test features are more advanced.

Information as to standards and common module usage by AIPS was not available for this review. If the costs savings associated with common modules and adherence to standards hold for the Boeing and the General Dynamics architecture, it would be difficult for AIPS to compete on recurring life cycle costs without using common modules and adhering to supported standards.

APPENDIX A

FUNCTIONAL DECOMPOSITION OF MPRAS REQUIREMENTS

Functional Decomposition of Boeing MPRAS Requirements

Context-Diagram;9
MPRAS

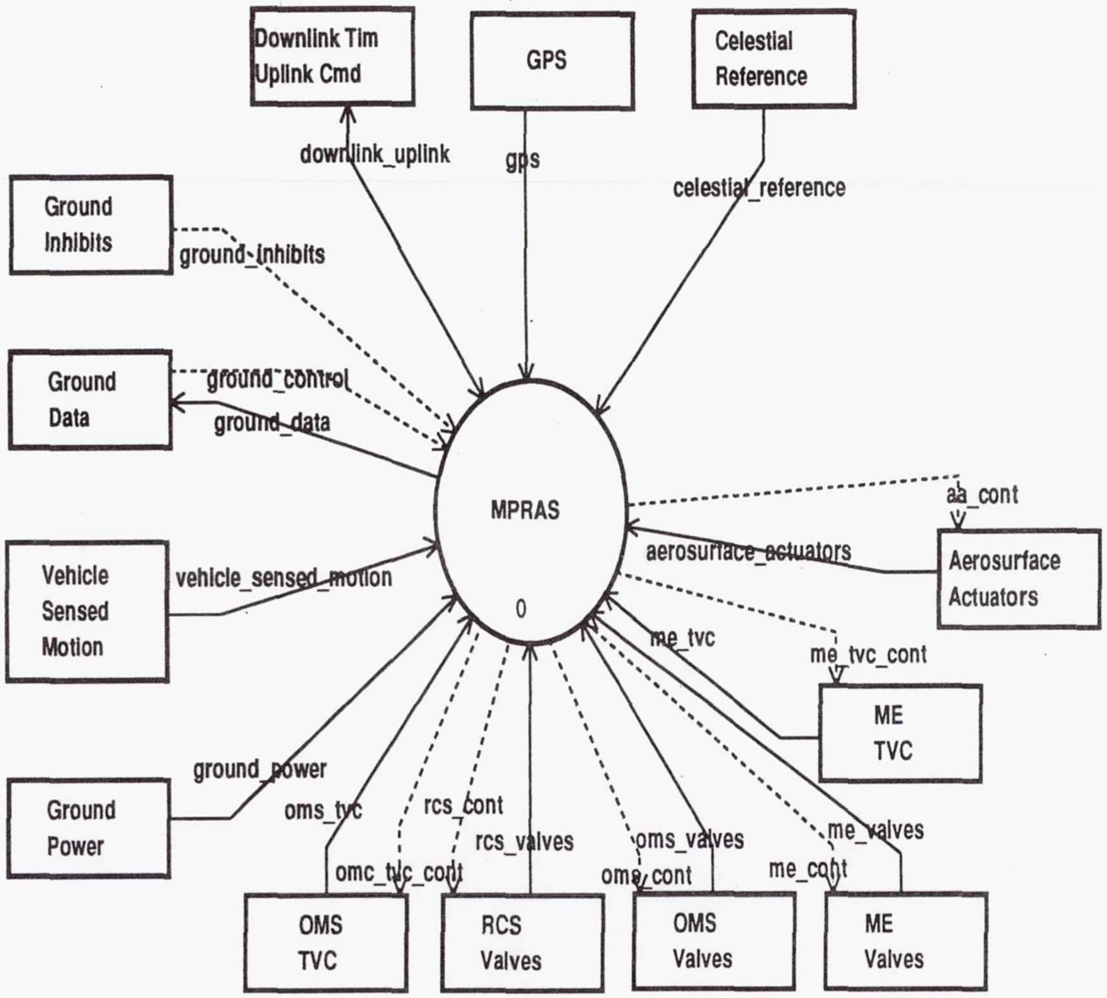


Figure 1.

0;4
MPRAS

" in " ground_power

" in " vehicle_sensed_motion

" out " ground_data

" in " ground_inhibits

" inout " downlink_uplink

" in " gps

" in " celestial_reference

" in " aerosurace_actuators

" in " me_tvc

" in " me_valves

" in " oms_valves

" in " rcs_valves

" in " oms_tvc

" in " ground_control

" out " omc_cont

" out " rcs_cont

" out " oms_tvc_cont

" out " me_cont

" out " me_tvc_cont

" out " aa_cont

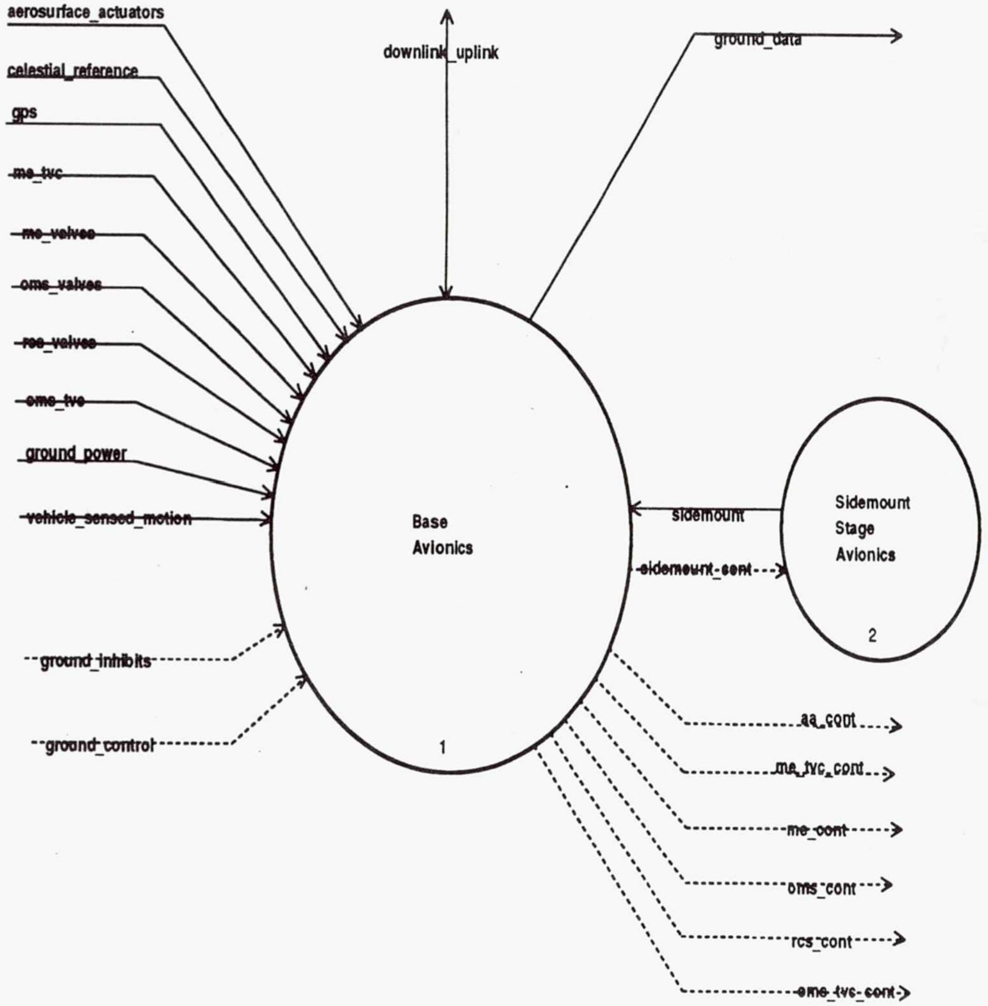


Figure 2.

1;7
Base Avionics

' in ' sidemount
' out ' sidemount_cont
' in ' gps
' in ' celestial_reference
' in ' me_tvc
' in ' me_valves
' in ' oms_valves
' in ' rcs_valves
' in ' oms_tvc
' in ' ground_power

' in ' aerosurface_actuators
' in ' ground_inhibits
' in ' ground_control
' out ' ground_data
' inout ' downlink_uplink
' out ' aa_cont
' out ' me_tvc_cont
' out ' me_cont
' out ' oms_cont
' out ' rcs_cont
' out ' oms_tvc_cont

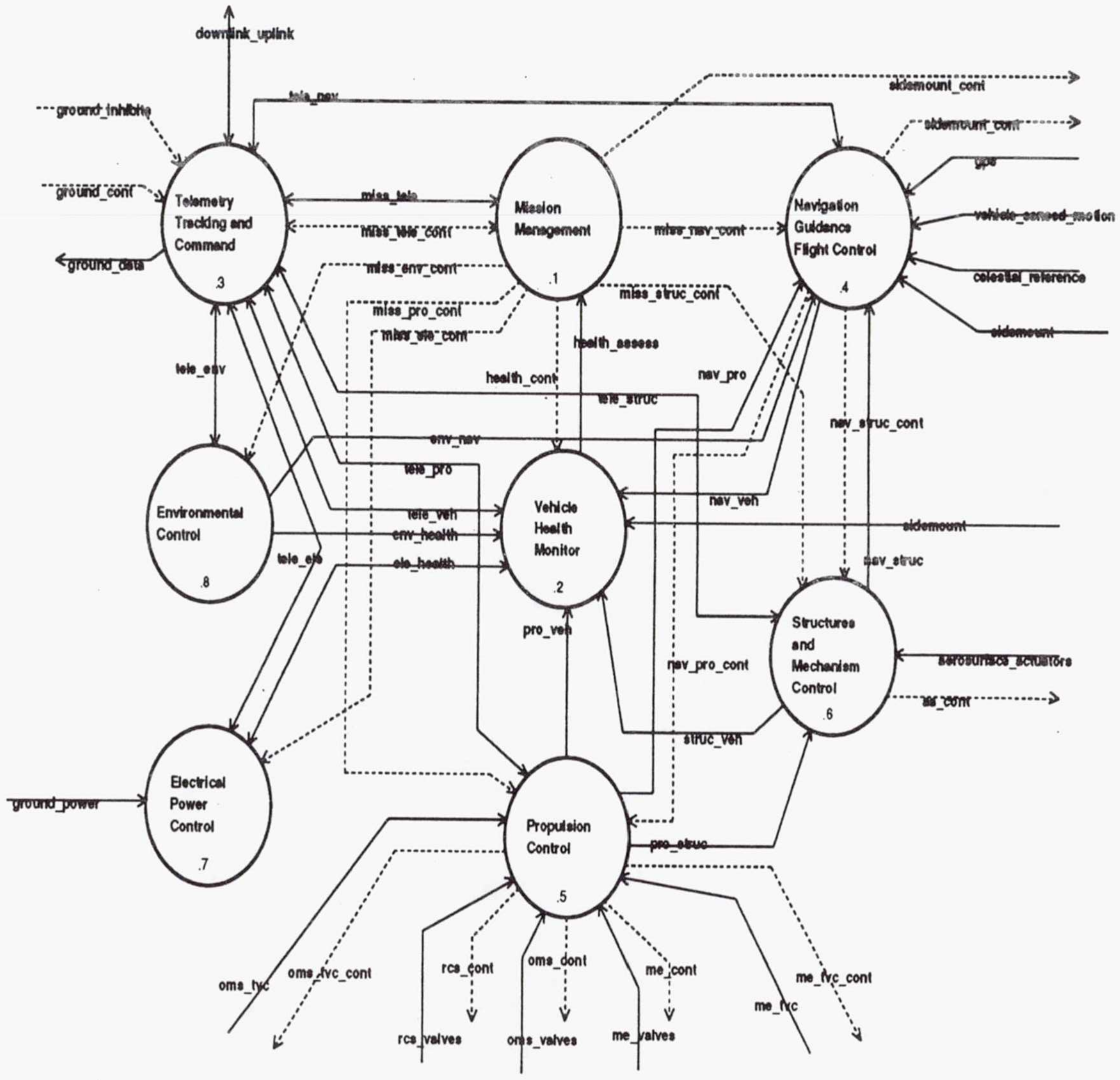


Figure 3.

1.1;5
Mission Management

* inout * miss_tele_cont

* out * miss_env_cont

* out * miss_nav_cont

* out * sidemount_cont

* out * miss_struc_cont

* out * miss_pro_cont

* in * health_assess

* out * health_cont

* out * miss_ele_cont

* inout * miss_tele

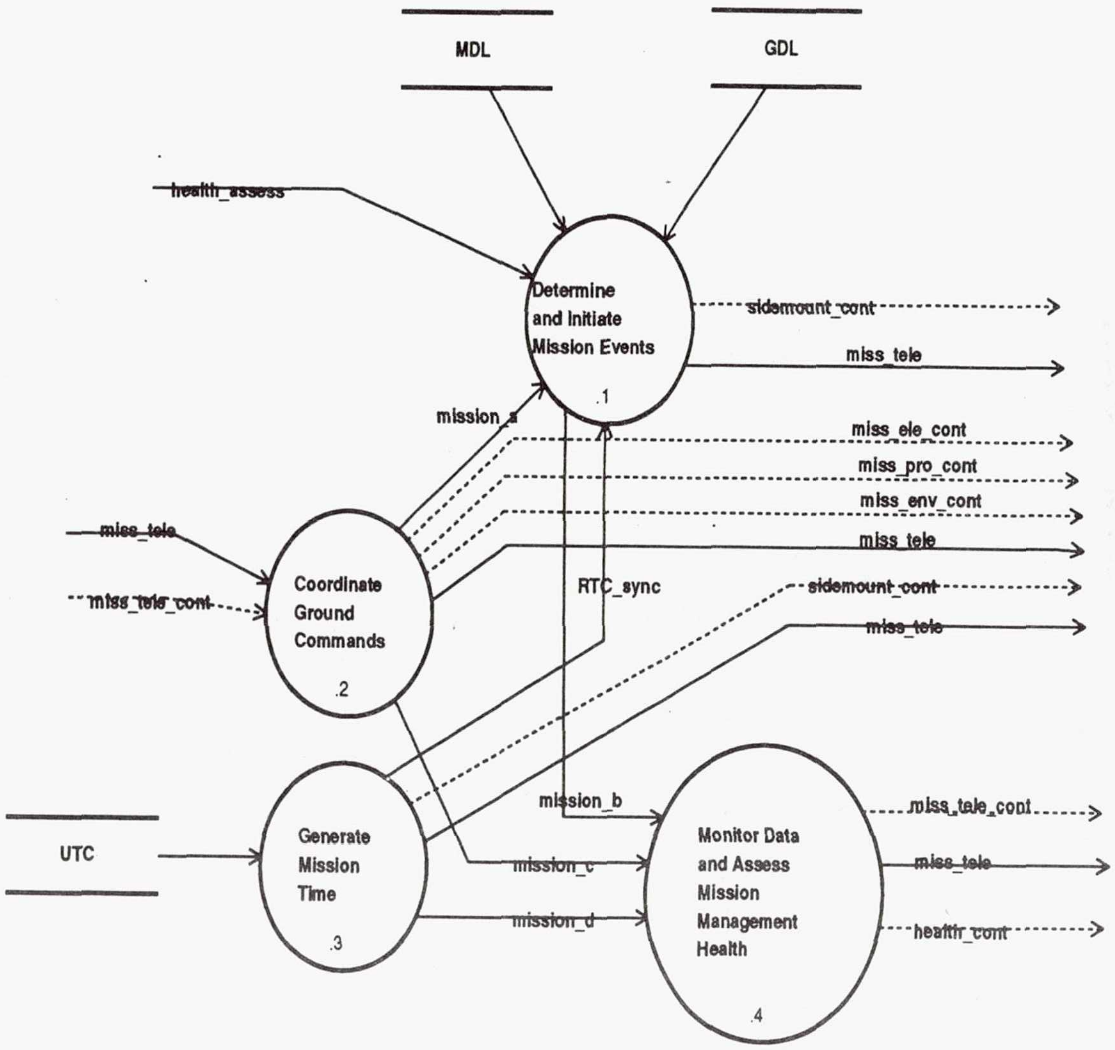


Figure 4.

1.2;1
Vehicle Health Monitor

* inout * tele_veh

* in * env_health

* inout * ele_health

* in * pro_veh

* in * struc_veh

* in * sidemount

* in * nav_veh

* out * health_assess

* in * health_cont

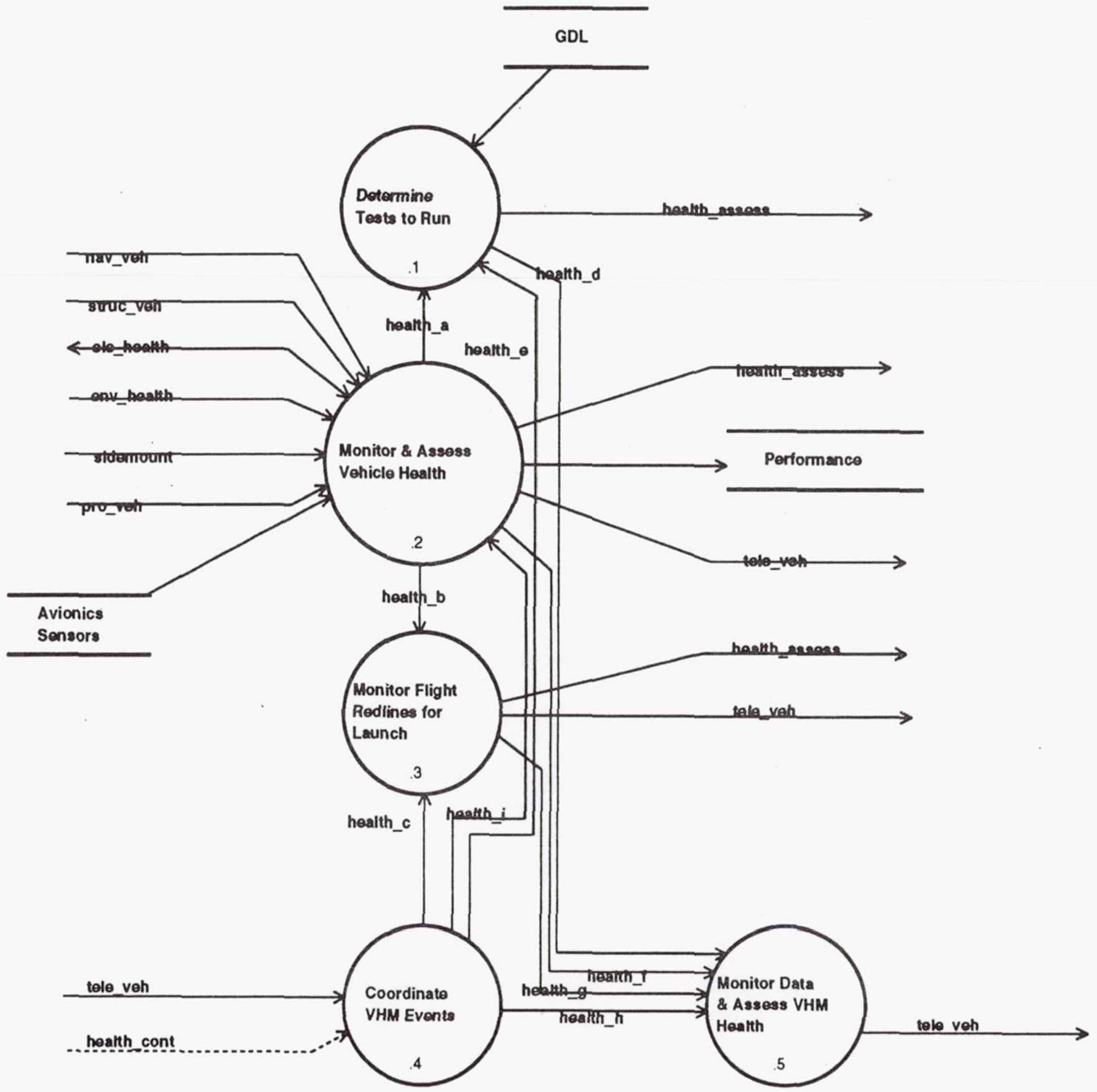


Figure 5.

1.3;1
Telemetry Tracking and Command

* inout * downlink_uplink

* in * ground_inhibits

* in * ground_cont

* out * ground_data

* inout * miss_tele_cont

* inout * tele_veh

* inout * tele_env

* inout * tele_ele

* inout * miss_tele

* inout * tele_nav

* inout * tele_pro

* inout * tele_struc

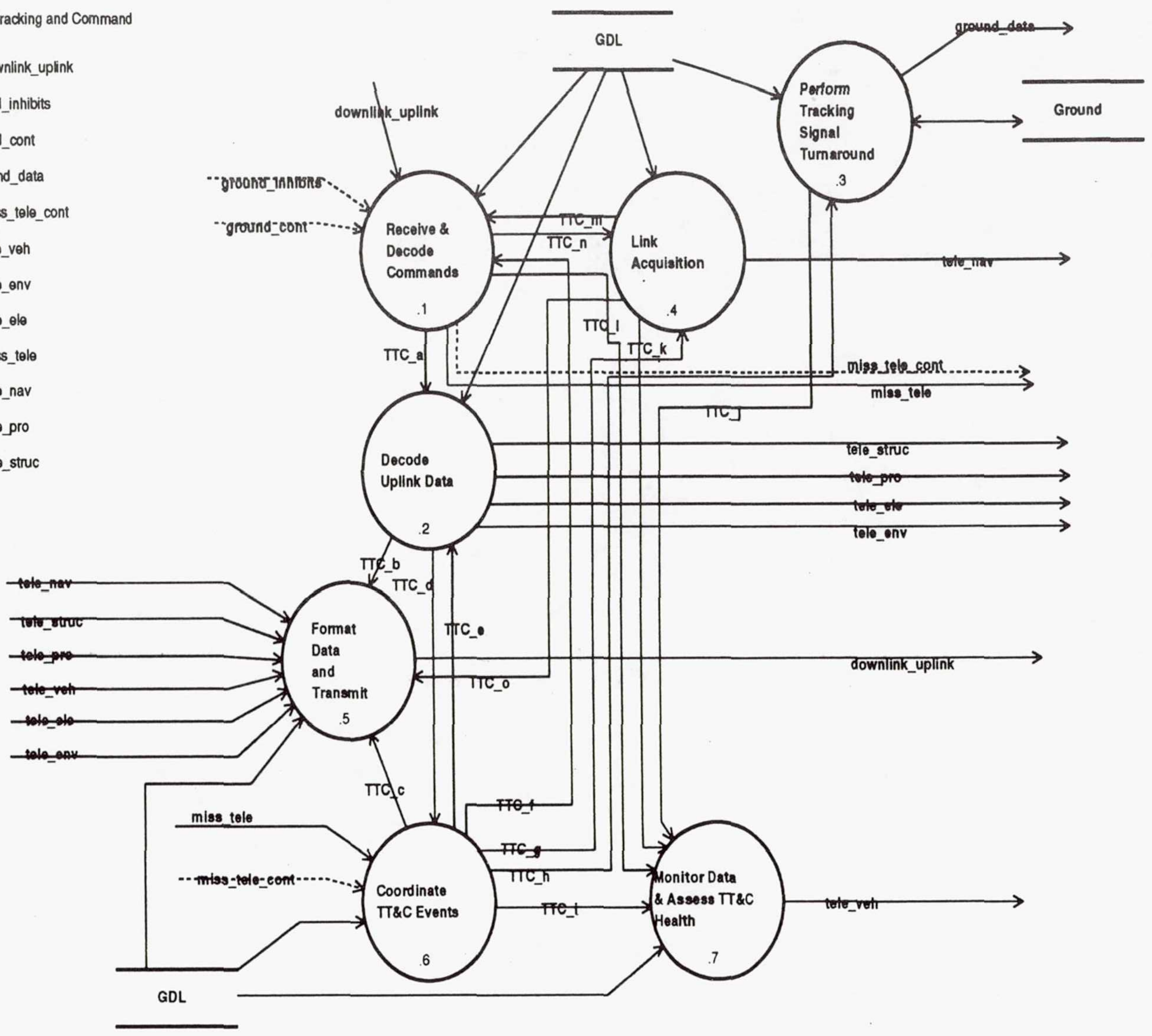


Figure 6.

1.4;5

Navigation Guidance Flight Control

* out * nav_miss

* in * miss_nav_cont

* out * nav_veh

* in * gps

* in * vehicle_sensed_motion

* in * celestial_reference

* out * sidemount_cont

* in * sidemount

* in * nav_struc

* out * nav_struc_cont

* out * nav_pro_cont

* in * nav_pro

* inout * tele_nav

* in * env_nav

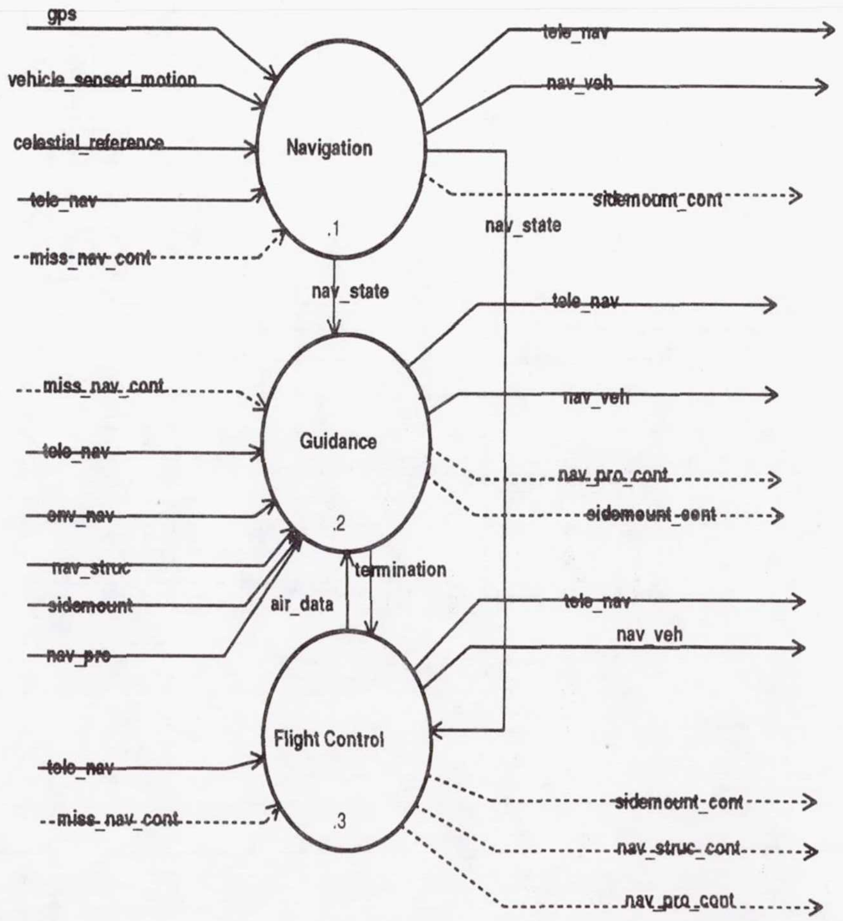


Figure 7.

1.4.1;3
Navigation

* in * gps

* in * vehicle_sensed_motion

* in * celestial_reference

* out * nav_state

* out * nav_state

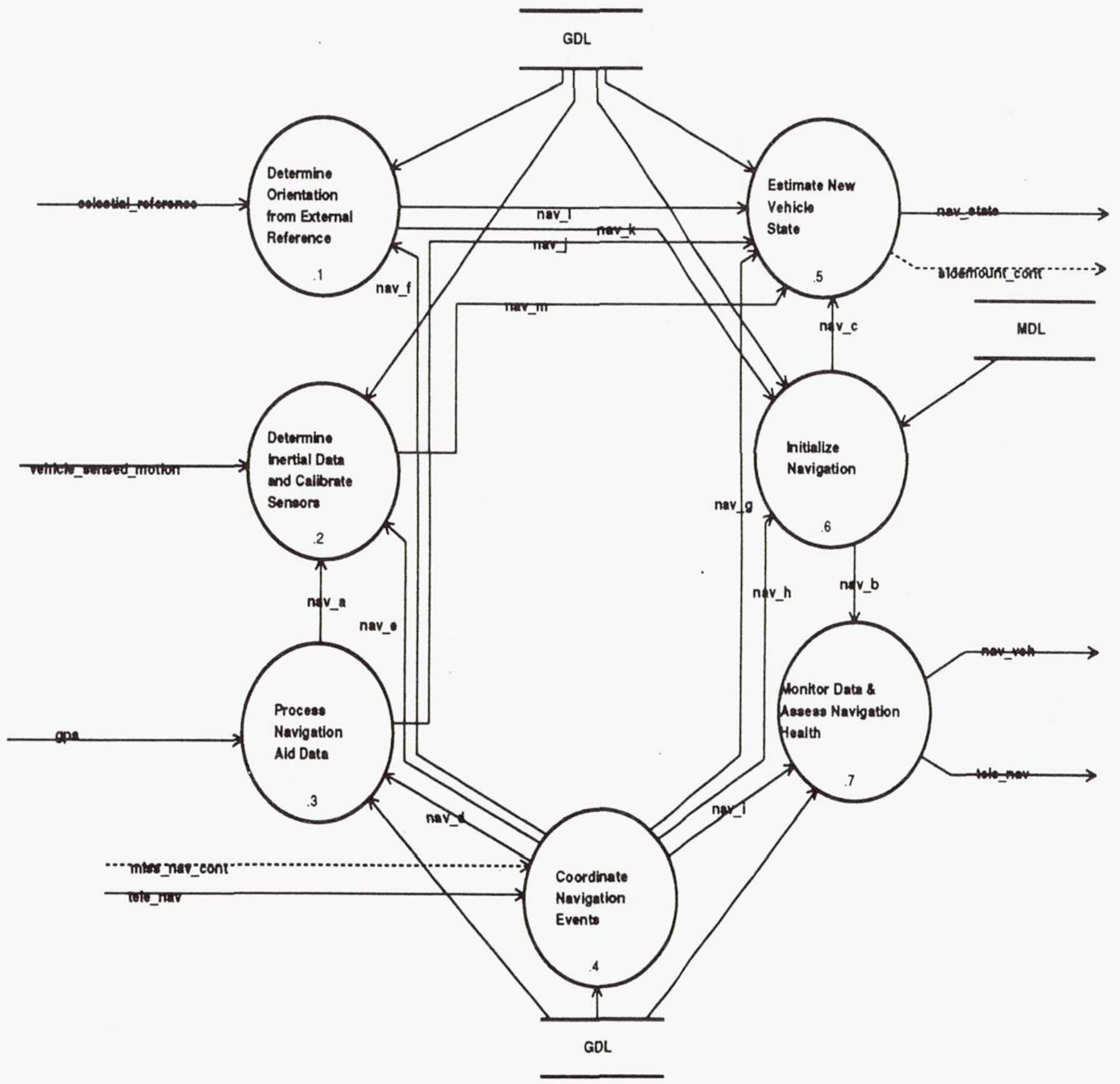
* out * tele_nav

* in * tele_nav

* in * miss_nav_cont

* out * sidemount_cont

* out * nav_veh



GDL
Determine Orientation from External Reference .1
Estimate New Vehicle State .5
Determine Inertial Data and Calibrate Sensors .2
Initialize Navigation .6
Process Navigation Aid Data .3
Monitor Data & Assess Navigation Health .7
Coordinate Navigation Events .4
MDL
GDL
celestial_reference
vehicle_sensed_motion
gps
miss_nav_cont
tele_nav
nav_state
sidemount_cont
nav_veh
tele_nav
nav_a
nav_b
nav_c
nav_d
nav_e
nav_f
nav_g
nav_h
nav_i
nav_j
nav_k
nav_l
nav_m

Figure 8.

1.4.2.2
Guidance

" in " miss_nav_cont

" in " nav_state

" in " air_data

" out " termination

" out " tele_nav

" in " tele_nav

" out " sidemount_cont

" out " nav_veh

" in " env_nav

" in " nav_struc

" in " sidemount

" in " nav_pro

" out " nav_pro_cont

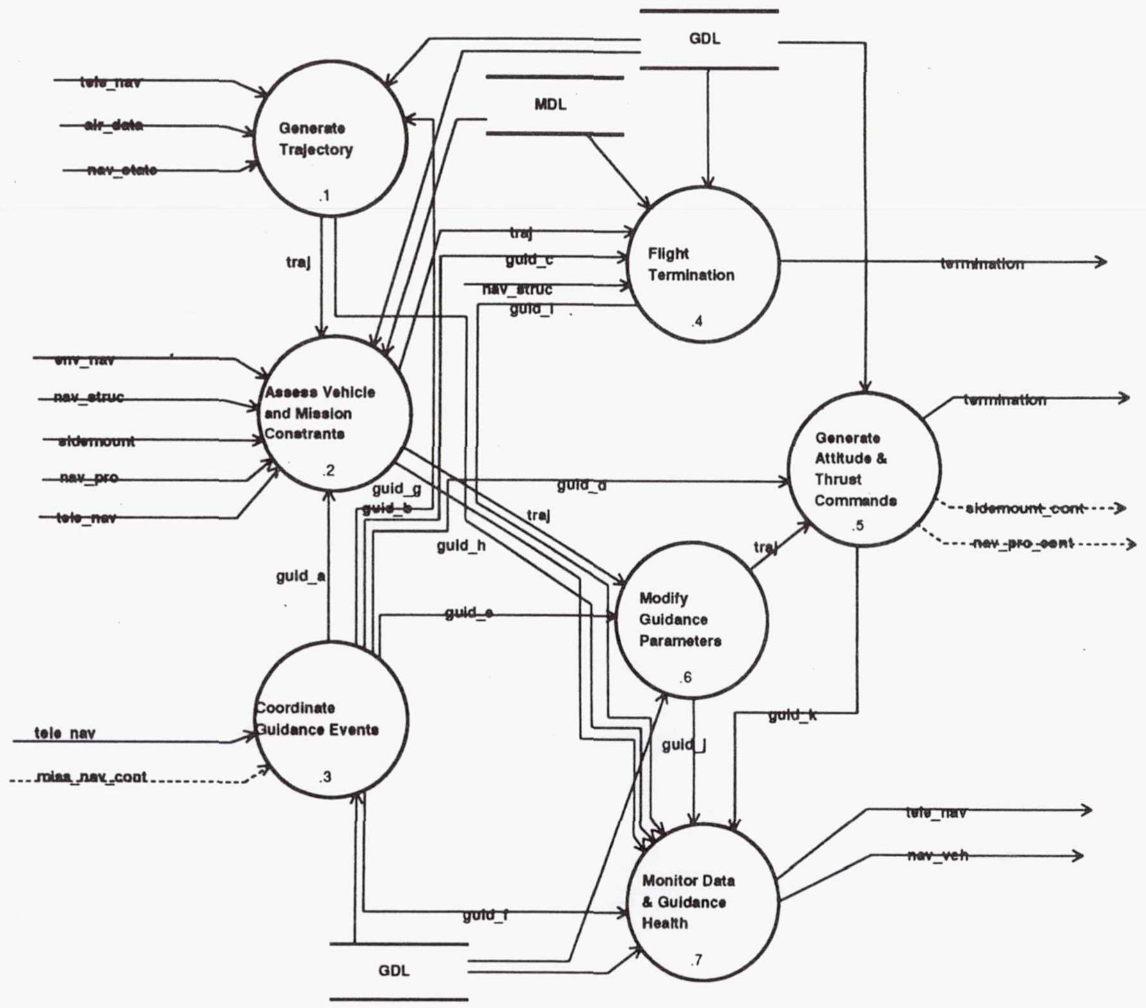


Figure 9.

1.4.3;3
Flight Control

* in * nav_state

* out * air_data

* in * termination

* out * nav_veh

* out * sidemount_cont

* out * tele_nav

* in * tele_nav

* out * nav_struc_cont

* out * nav_pro_cont

* in * miss_nav_cont

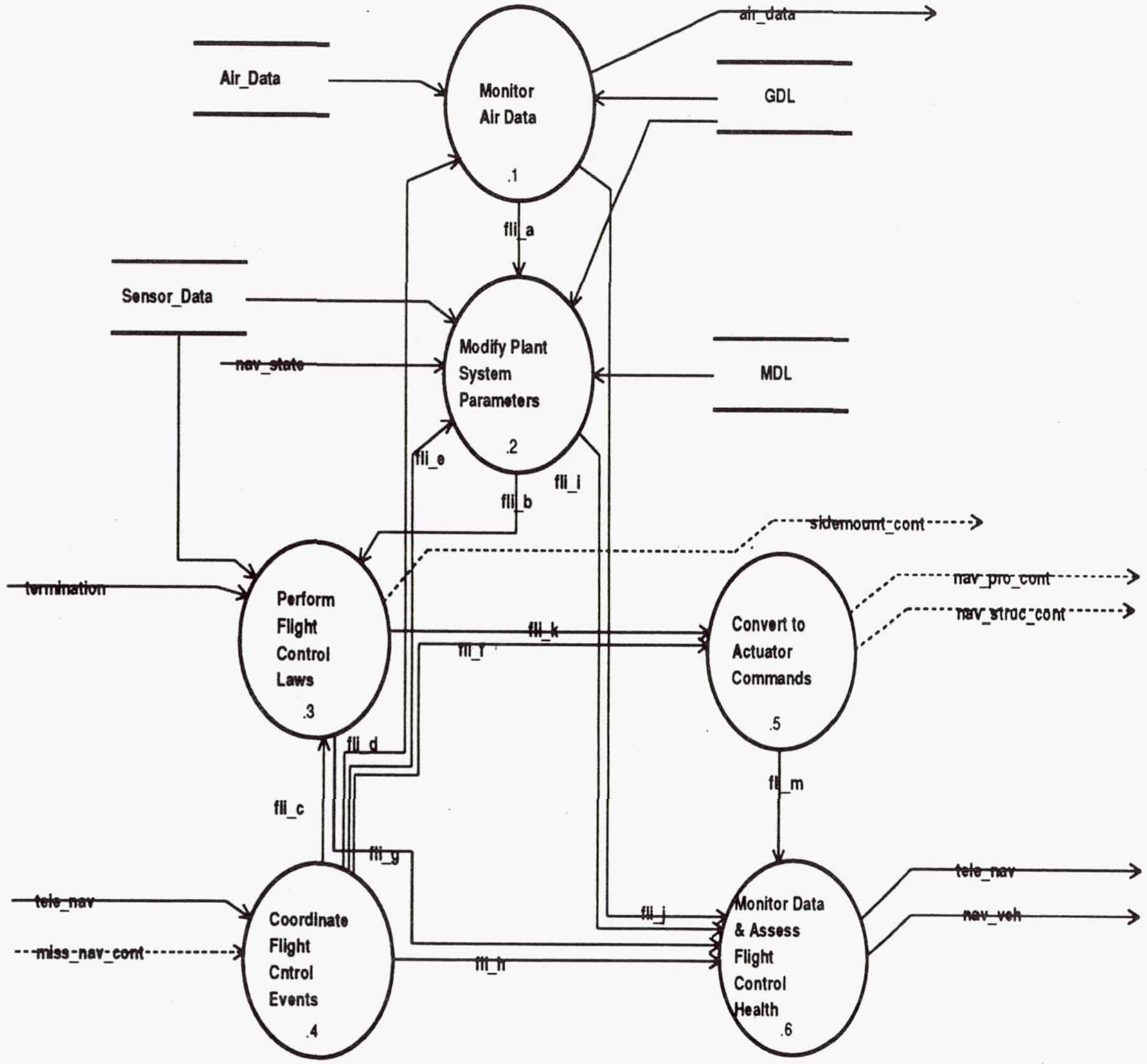


Figure 10.

1.5;3
Propulsion Control

* out * pro_veh

* out * me_tvc_cont

* out * me_cont

* out * oms_cont

* out * rcs_cont

* out * oms_tvc_cont

* in * me_tvc

* in * me_valves

* in * oms_valves

* in * rcs_valves

* in * oms_tvc

* in * miss_pro_cont

* in * nav_pro_cont

* out * nav_pro

* inout * tele_pro

* out * pro_struc

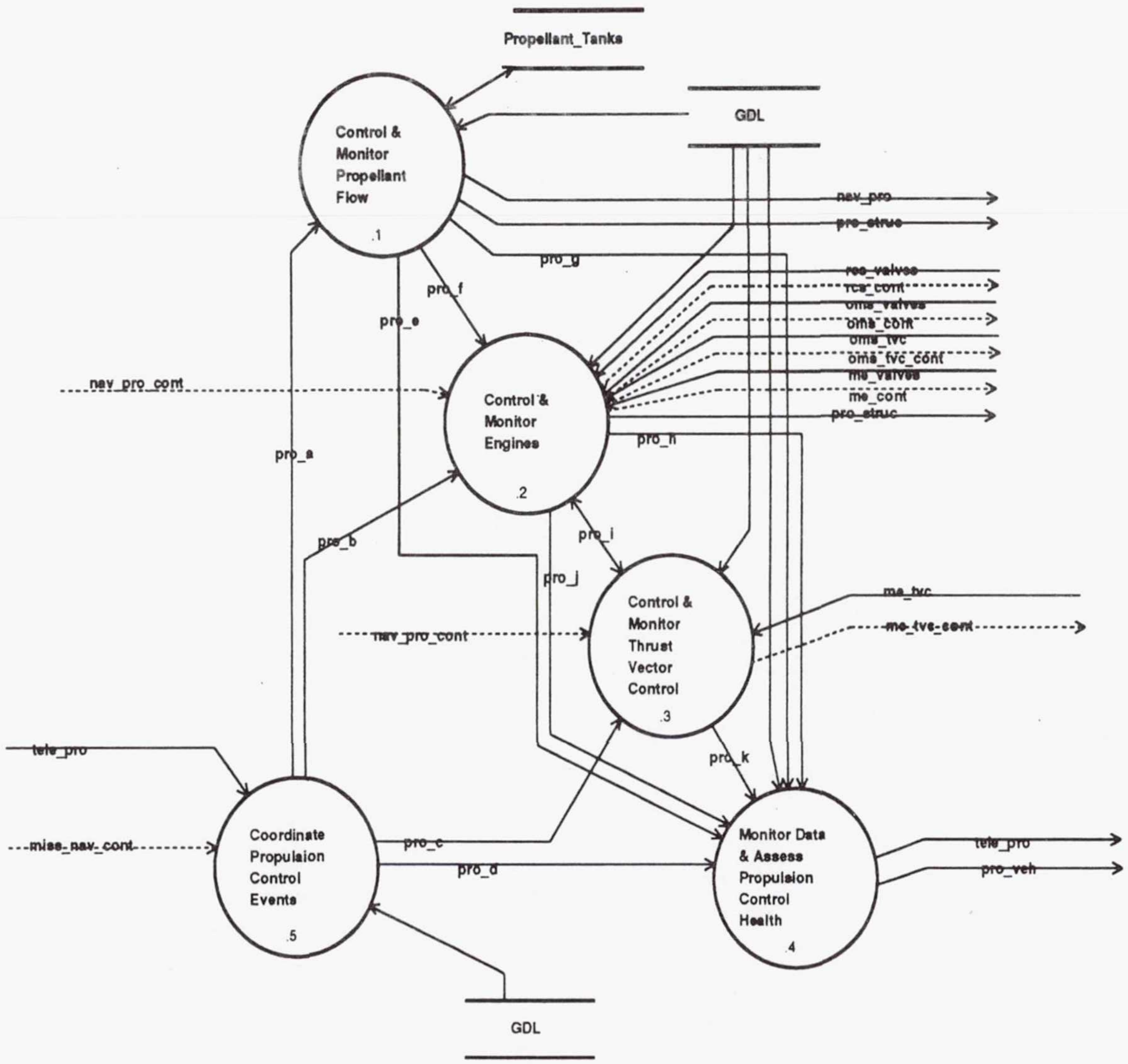


Figure 11.

1.6;1
Structures and Mechanism Control

* out * struc_veh

* out * nav_struc

* in * nav_struc_cont

* in * aerosurface_actuators

* out * aa_cont

* in * miss_struc_cont

* inout * tele_struc

* in * pro_struc

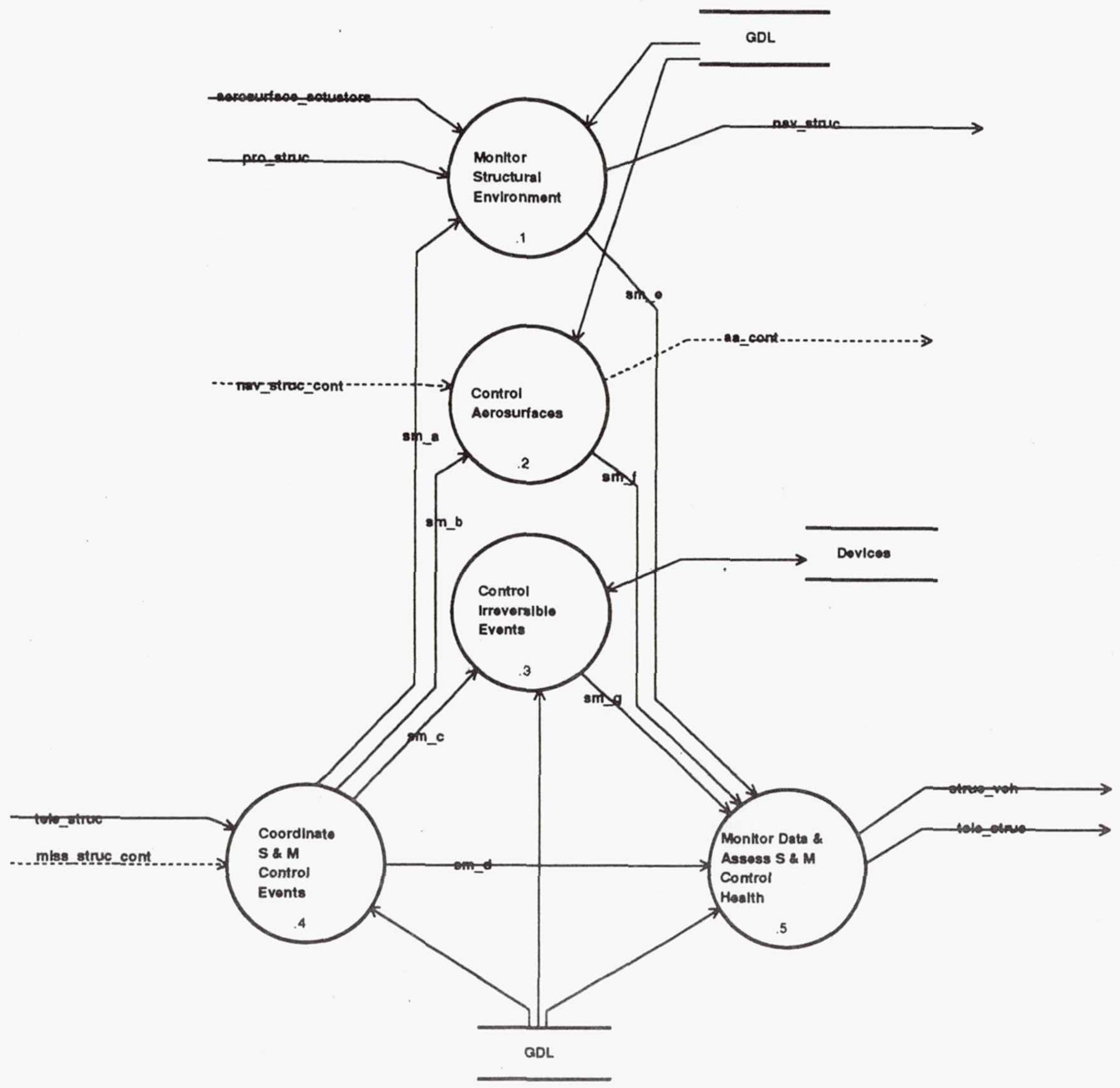


Figure 12.

1.7.3
Electrical Power Control

* out * ele_health

* in * miss_ele_cont

* in * ground_power

* inout * tele_ele

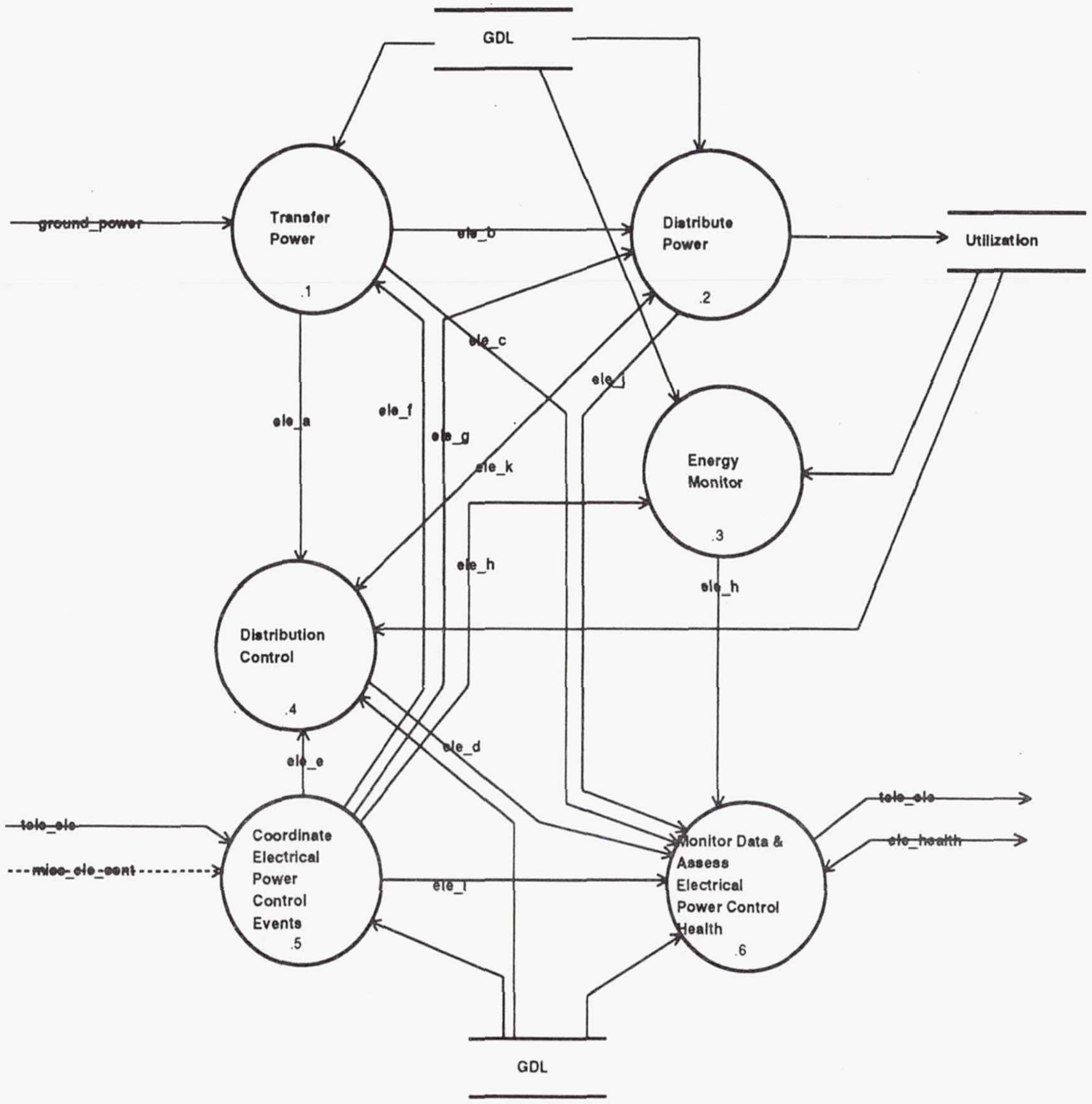


Figure 13.

1.8;1
Environmental Control

* in * miss_env_cont

* out * env_health

* inout * tele_env

* out * env_nav

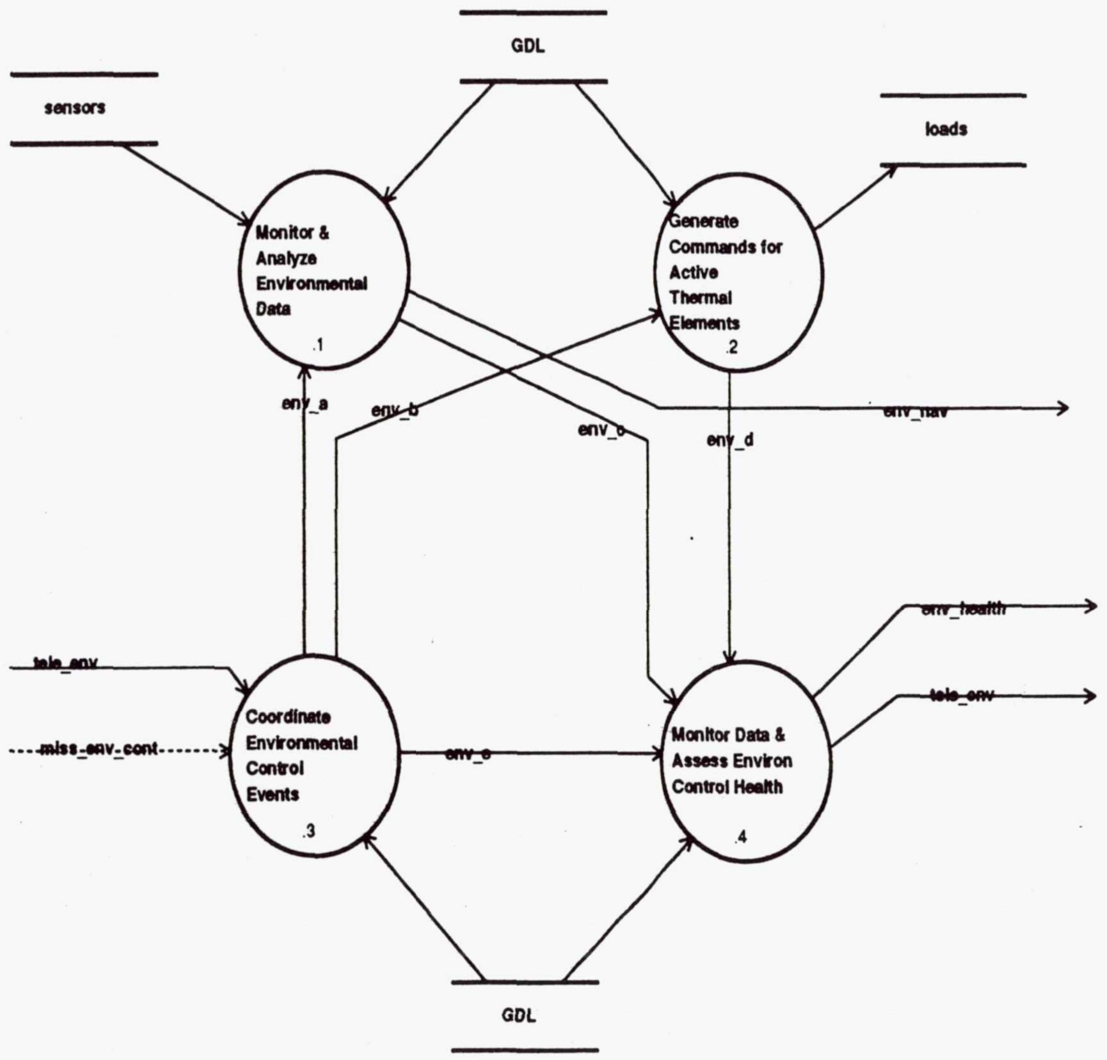
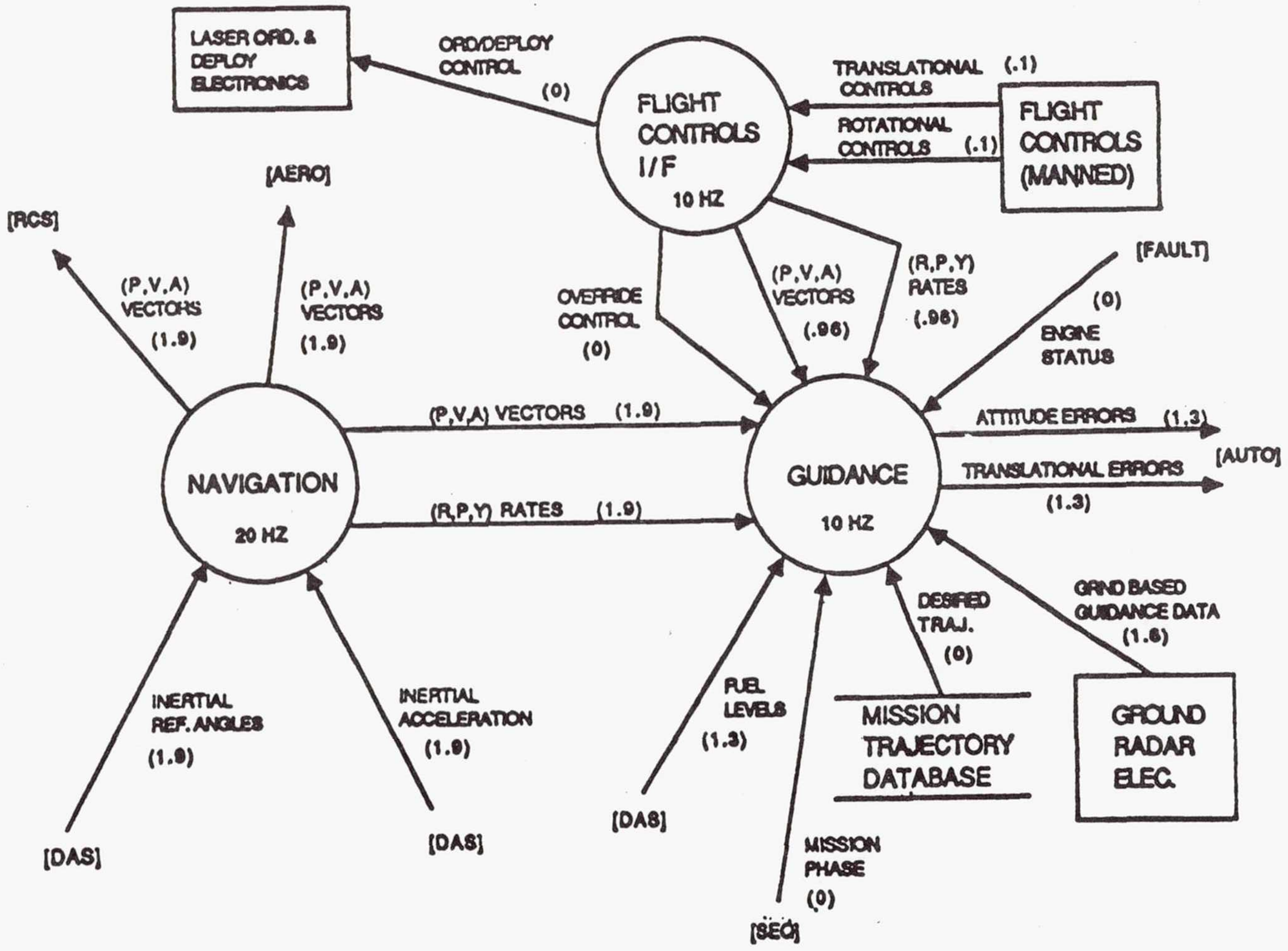
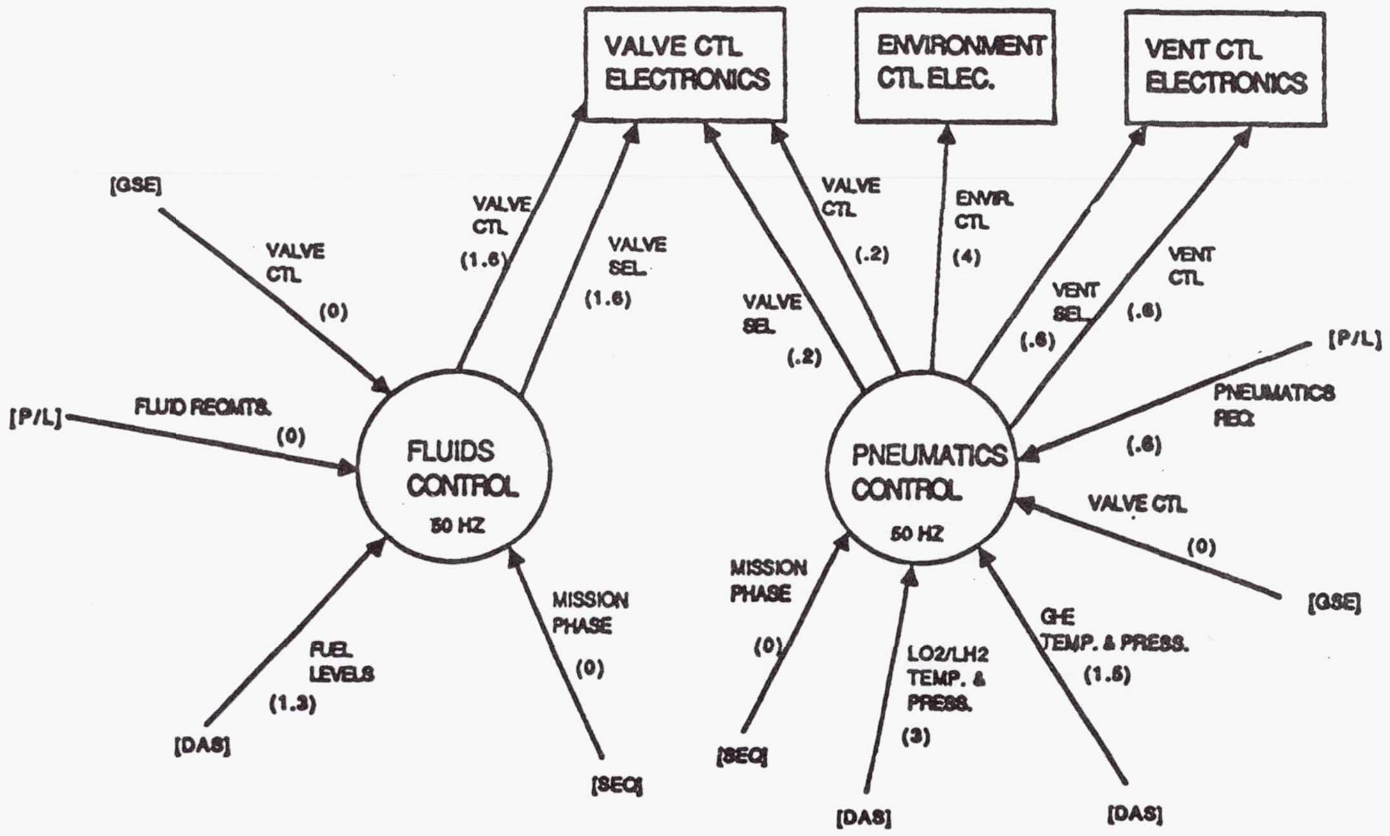
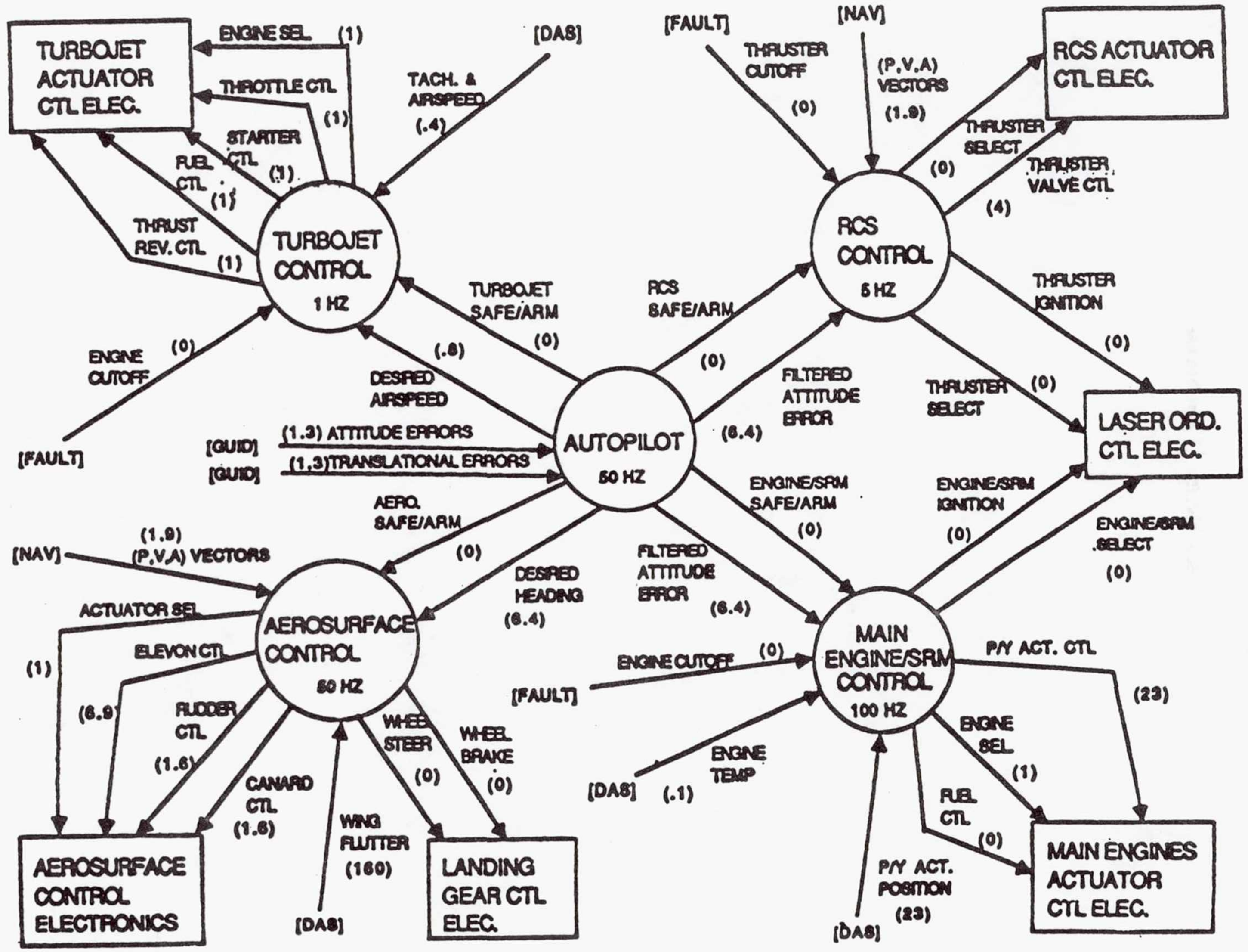

GDL
sensors
loads
Monitor & Analyze Environmental Data .1
Generate Commands for Active Thermal Elements .2
Coordinate Environmental Control Events .3
Monitor Data & Assess Environ Control Health .4
env_a
env_b
env_c
env_d
env_e
env_nav
env_health
tele_env
tele_env
miss_env_cont
GDL

Figure 14.

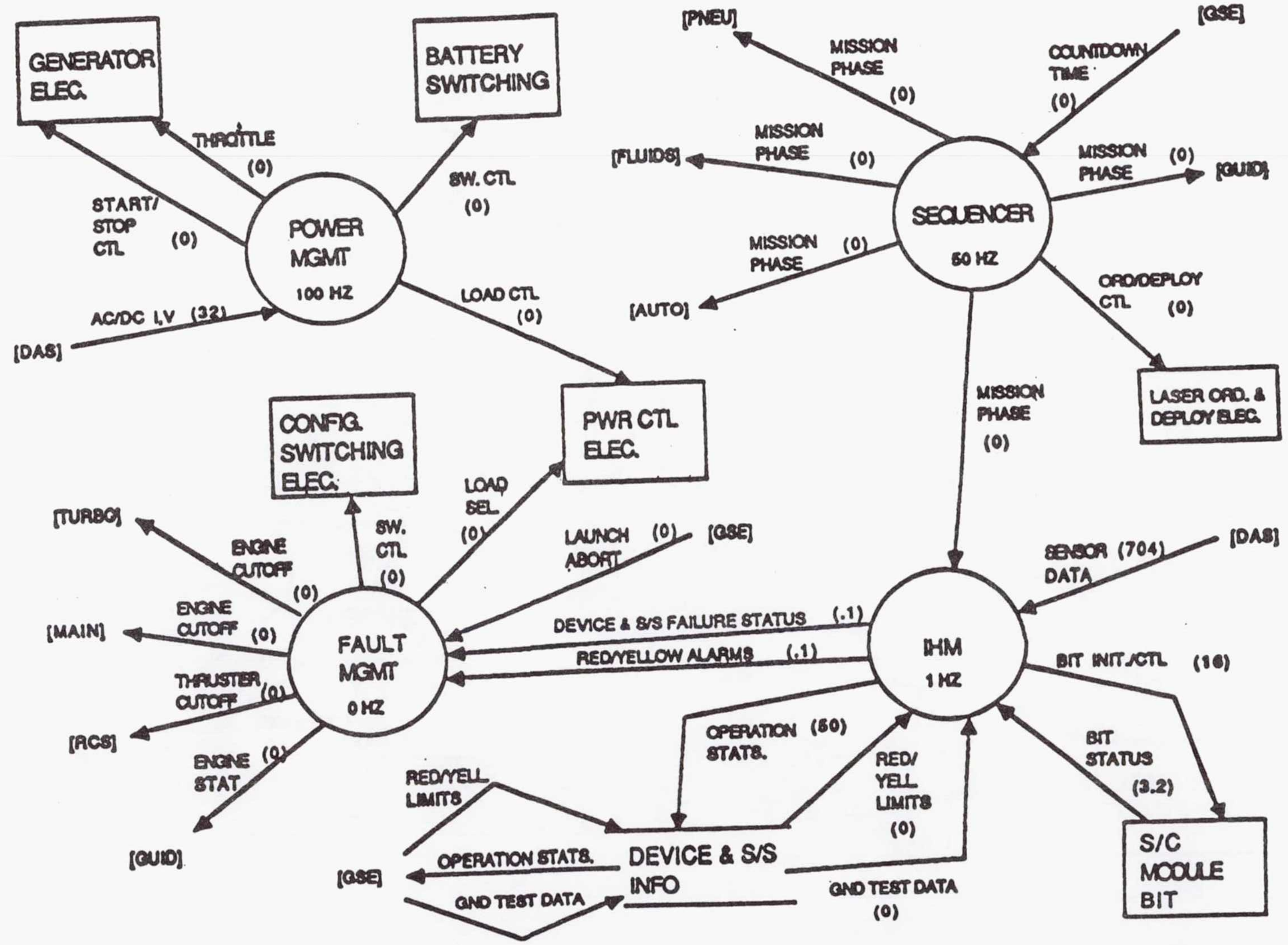
Functional Decomposition of General Dynamics MPRAS Requirements

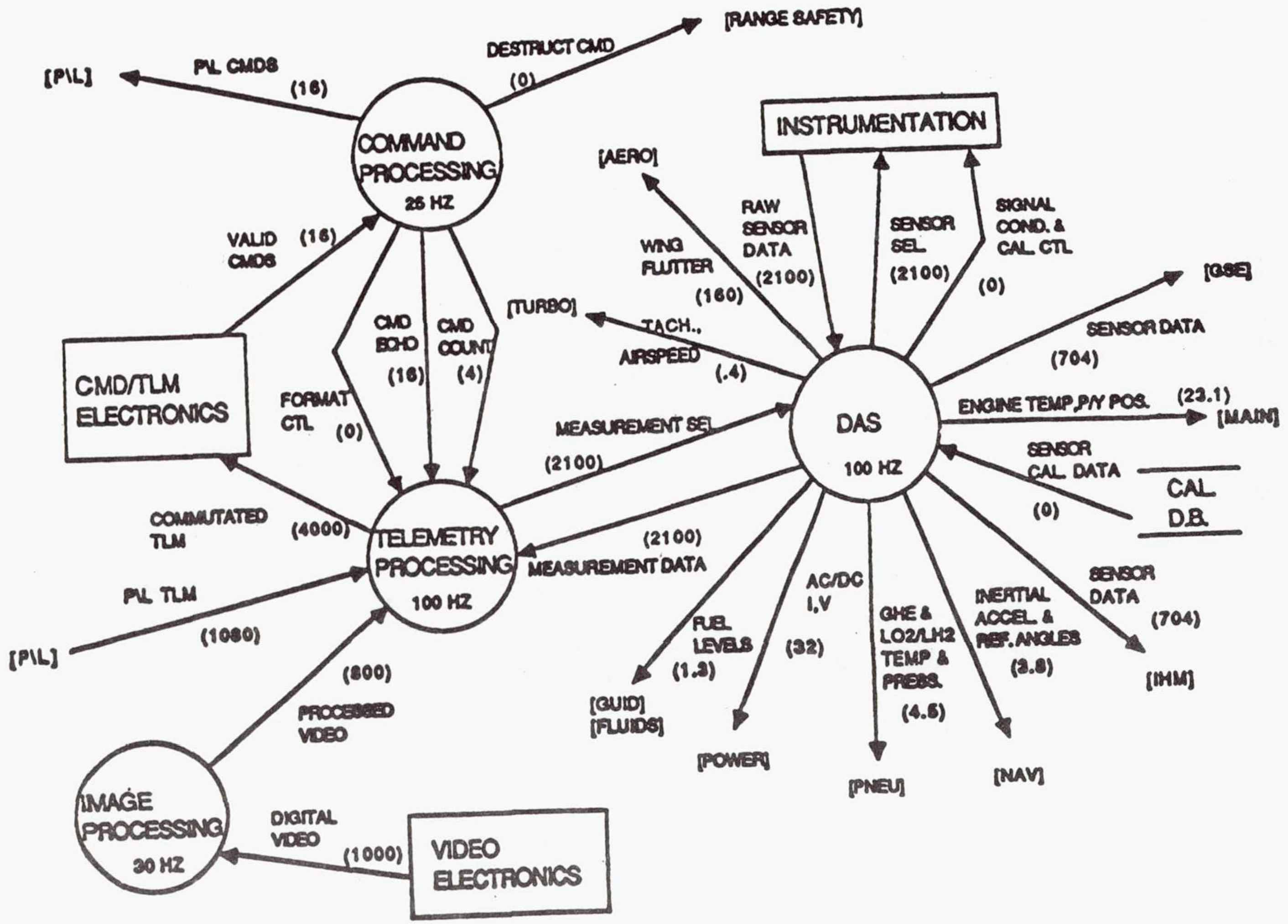

LASER ORD. & DEPLOY ELECTRONICS
ORD/DEPLOY CONTROL (0)
FLIGHT CONTROLS I/F 10 HZ
TRANSLATIONAL CONTROLS (.1)
ROTATIONAL CONTROLS (.1)
FLIGHT CONTROLS (MANNED)
[AERO]
[RCS]
(P,V,A) VECTORS (1.9)
(P,V,A) VECTORS (1.9)
OVERRIDE CONTROL (0)
(P,V,A) VECTORS (.96)
(R,P,Y) RATES (.96)
[FAULT]
(0)
ENGINE STATUS
(P,V,A) VECTORS (1.9)
ATTITUDE ERRORS (1.3)
[AUTO]
NAVIGATION 20 HZ
GUIDANCE 10 HZ
TRANSLATIONAL ERRORS (1.3)
(R,P,Y) RATES (1.9)
DESIRED TRAJ. (0)
GRND BASED GUIDANCE DATA (1.6)
INERTIAL REF. ANGLES (1.9)
INERTIAL ACCELERATION (1.9)
FUEL LEVELS (1.3)
MISSION TRAJECTORY DATABASE
GROUND RADAR ELEC.
[DAS]
[DAS]
[DAS]
MISSION PHASE (0)
[SEQ]


VALVE CTL ELECTRONICS
ENVIRONMENT CTL ELEC.
VENT CTL ELECTRONICS
FLUIDS CONTROL 50 HZ
PNEUMATICS CONTROL 50 HZ
[GSE]
VALVE CTL (0)
[P/L]
FLUID REQMTS. (0)
VALVE CTL (1.6)
VALVE SEL (1.6)
[DAS]
FUEL LEVELS (1.3)
MISSION PHASE (0)
[SEQ]
VALVE SEL (.2)
VALVE CTL (.2)
ENVIR. CTL (4)
VENT SEL (.6)
VENT CTL (.6)
[P/L]
PNEUMATICS REQ (.6)
VALVE CTL (0)
[GSE]
MISSION PHASE (0)
[SEQ]
LO2/LH2 TEMP. & PRESS. (3)
[DAS]
GHE TEMP. & PRESS. (1.5)
[DAS]

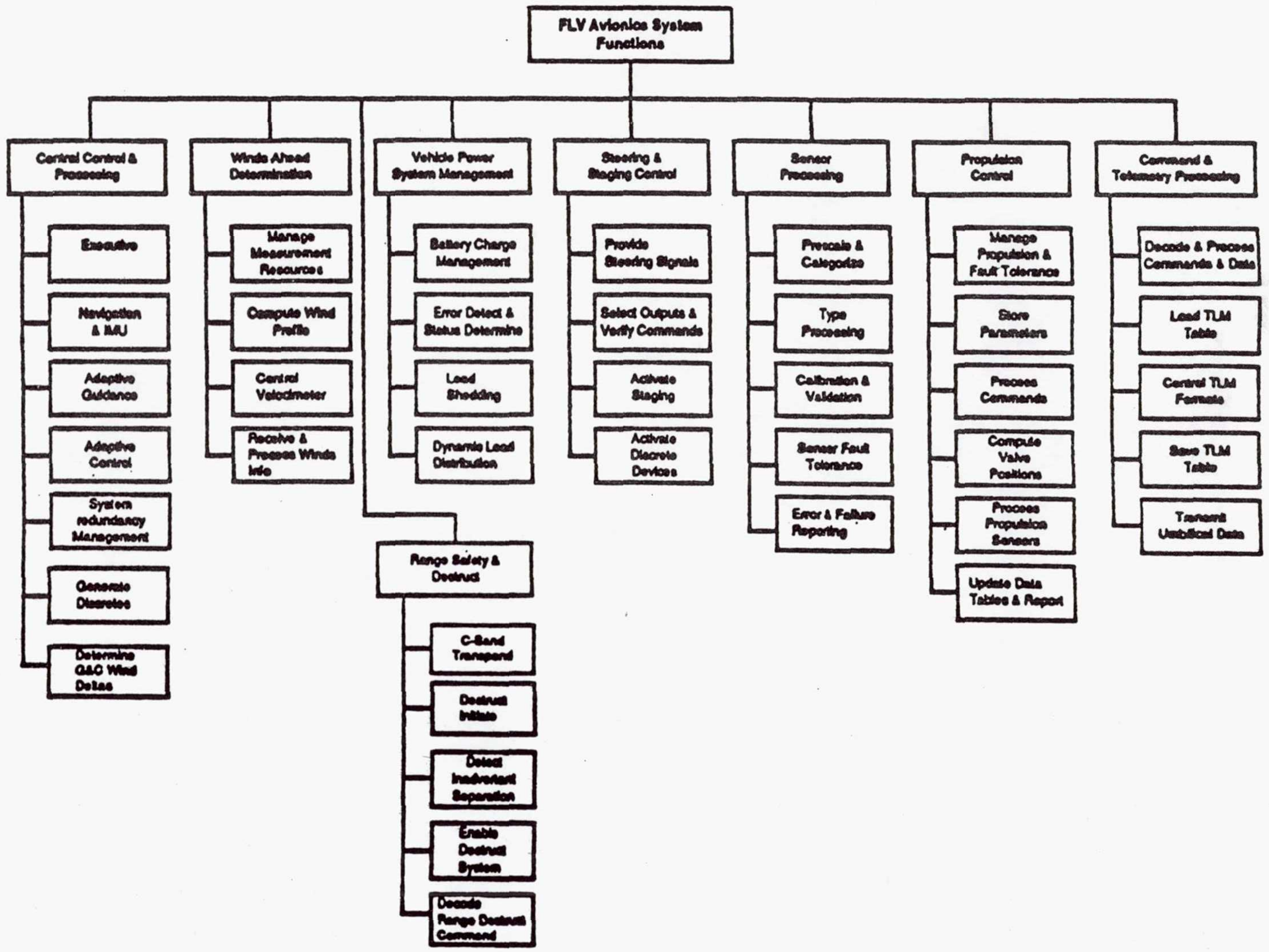


TURBOJET ACTUATOR CTL ELEC.
ENGINE SEL. (1)
THROTTLE CTL (1)
STARTER CTL (1)
FUEL CTL (1)
THRUST REV. CTL (1)
[DAS]
TACH. & AIRSPEED (.4)
TURBOJET CONTROL 1 HZ
ENGINE CUTOFF (0)
[FAULT]
TURBOJET SAFE/ARM (0)
DESIRED AIRSPEED (.8)
[GUID] (1.3) ATTITUDE ERRORS
[GUID] (1.3) TRANSLATIONAL ERRORS
AUTOPILOT 50 HZ
[FAULT]
THRUSTER CUTOFF (0)
[NAV]
(P,V,A) VECTORS (1.9)
RCS ACTUATOR CTL ELEC.
THRUSTER SELECT (0)
THRUSTER VALVE CTL (4)
RCS CONTROL 5 HZ
RCS SAFE/ARM (0)
FILTERED ATTITUDE ERROR (6.4)
THRUSTER IGNITION (0)
THRUSTER SELECT (0)
LASER ORD. CTL ELEC.
ENGINE/SRM SAFE/ARM (0)
ENGINE/SRM IGNITION (0)
ENGINE/SRM SELECT (0)
AERO. SAFE/ARM (0)
[NAV]
(1.9) (P,V,A) VECTORS
AEROSURFACE CONTROL 50 HZ
ACTUATOR SEL (1)
ELEVON CTL (6.9)
RUDDER CTL (1.6)
CANARD CTL (1.6)
WING FLUTTER (160)
[DAS]
WHEEL STEER (0)
WHEEL BRAKE (0)
AEROSURFACE CONTROL ELECTRONICS
LANDING GEAR CTL ELEC.
DESIRED HEADING (6.4)
FILTERED ATTITUDE ERROR (6.4)
MAIN ENGINE/SRM CONTROL 100 HZ
[FAULT]
ENGINE CUTOFF (0)
[DAS]
ENGINE TEMP (.1)
P/Y ACT. CTL (23)
ENGINE SEL (1)
FUEL CTL (0)
P/Y ACT. POSITION (23)
[DAS]
MAIN ENGINES ACTUATOR CTL ELEC.


GENERATOR ELEC.
BATTERY SWITCHING
THROTTLE (0)
START/ STOP CTL (0)
SW. CTL (0)
POWER MGMT 100 HZ
AC/DC I,V (32)
[DAS]
LOAD CTL (0)
[PNEU]
MISSION PHASE (0)
[GSE]
COUNTDOWN TIME (0)
[FLUIDS]
MISSION PHASE (0)
MISSION PHASE (0)
[GUID]
SEQUENCER 50 HZ
MISSION PHASE (0)
[AUTO]
ORD/DEPLOY CTL (0)
LASER ORD. & DEPLOY ELEC.
MISSION PHASE (0)
CONFIG. SWITCHING ELEC.
PWR CTL ELEC.
LOAD SEL. (0)
SW. CTL (0)
[TURBO]
ENGINE CUTOFF (0)
LAUNCH ABORT (0)
[GSE]
[DAS]
SENSOR (704) DATA
[MAIN]
ENGINE CUTOFF (0)
DEVICE & S/S FAILURE STATUS (.1)
RED/YELLOW ALARMS (.1)
FAULT MGMT 0 HZ
IHM 1 HZ
THRUSTER CUTOFF (0)
[RCS]
BIT INIT./CTL (16)
ENGINE STAT (0)
[GUID]
OPERATION (50) STATS.
RED/YELL LIMITS
RED/ YELL LIMITS (0)
BIT STATUS (3.2)
[GSE]
OPERATION STATS.
DEVICE & S/S INFO
GND TEST DATA
GND TEST DATA (0)
S/C MODULE BIT


[RANGE SAFETY]
DESTRUCT CMD
(0)
[P\L]
P\L CMDS
(16)
COMMAND PROCESSING
25 HZ
INSTRUMENTATION
[AERO]
RAW SENSOR DATA
(2100)
SENSOR SEL.
(2100)
SIGNAL COND. & CAL. CTL
(0)
WING FLUTTER
(160)
[GSE]
SENSOR DATA
(704)
VALID CMDS
(16)
CMD ECHO
(16)
CMD COUNT
(4)
[TURBO]
TACH., AIRSPEED
(.4)
CMD/TLM ELECTRONICS
FORMAT CTL
(0)
MEASUREMENT SEL
(2100)
DAS
100 HZ
ENGINE TEMP,P/Y POS.
(23.1)
[MAIN]
SENSOR CAL. DATA
(0)
CAL. D.B.
COMMUTATED TLM
(4000)
TELEMETRY PROCESSING
100 HZ
(2100)
MEASUREMENT DATA
P\L TLM
(1080)
[P\L]
AC/DC I,V
(32)
FUEL LEVELS
(1.3)
GHE & LO2/LH2 TEMP & PRESS.
(4.5)
INERTIAL ACCEL. & REF. ANGLES
(3.8)
SENSOR DATA
(704)
[IHM]
[GUID]
[FLUIDS]
[POWER]
[PNEU]
[NAV]
(800)
PROCESSED VIDEO
IMAGE PROCESSING
30 HZ
DIGITAL VIDEO
(1000)
VIDEO ELECTRONICS

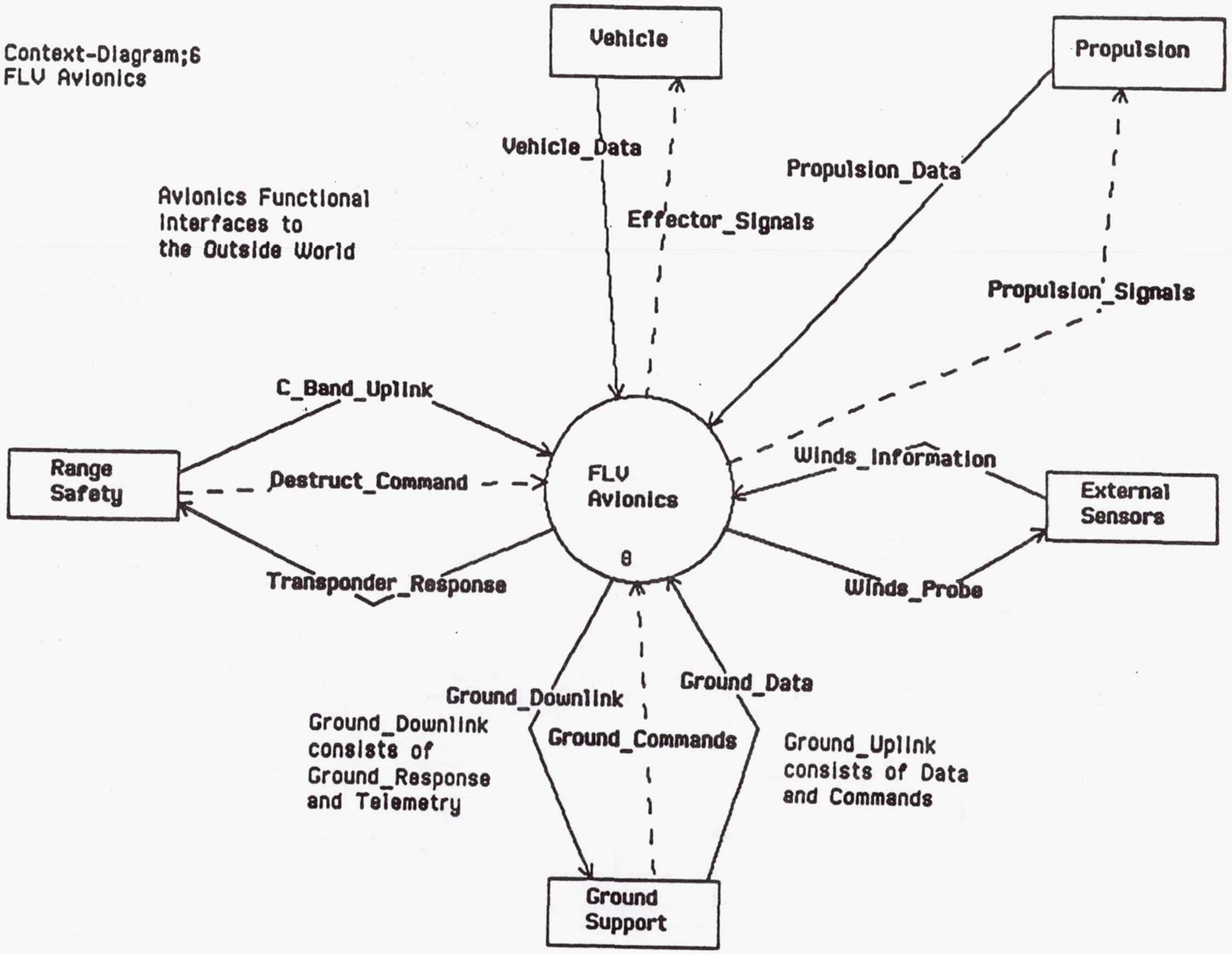
Functional Decomposition of Martin Marietta MPRAS Requirements


FLV Avionics System Functions
Central Control & Processing
Executive
Navigation & IMU
Adaptive Guidance
Adaptive Control
System redundancy Management
Generate Discretes
Determine G&C Wind Deltas
Winds Ahead Determination
Manage Measurement Resources
Compute Wind Profile
Control Velocimeter
Receive & Process Winds Info
Vehicle Power System Management
Battery Charge Management
Error Detect & Status Determine
Load Shedding
Dynamic Load Distribution
Range Safety & Destruct
C-Band Transpond
Destruct Initiate
Detect Inadvertent Separation
Enable Destruct System
Decode Range Destruct Command
Steering & Staging Control
Provide Steering Signals
Select Outputs & Verify Commands
Activate Staging
Activate Discrete Devices
Sensor Processing
Prescale & Categorize
Type Processing
Calibration & Validation
Sensor Fault Tolerance
Error & Failure Reporting
Propulsion Control
Manage Propulsion & Fault Tolerance
Store Parameters
Process Commands
Compute Valve Positions
Process Propulsion Sensors
Update Data Tables & Report
Command & Telemetry Processing
Decode & Process Commands & Data
Load TLM Table
Control TLM Formats
Save TLM Table
Transmit Umbilical Data

DOU/D18D-083/FLV FUNCTIONS-[illegible]

Context-Diagram;6
FLV Avionics

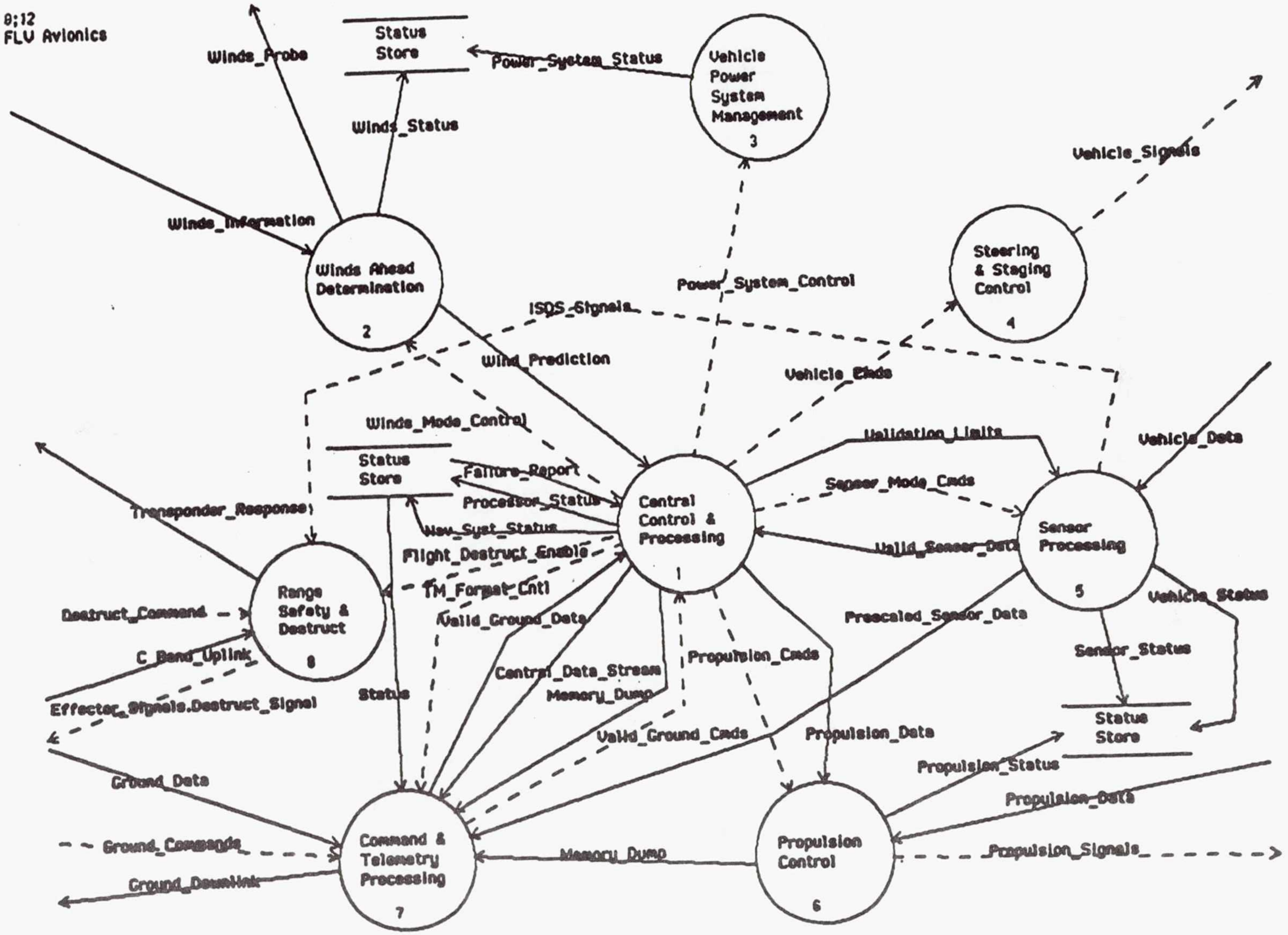
Avionics Functional
Interfaces to
the Outside World



Ground_Downlink consists of Ground_Response and Telemetry

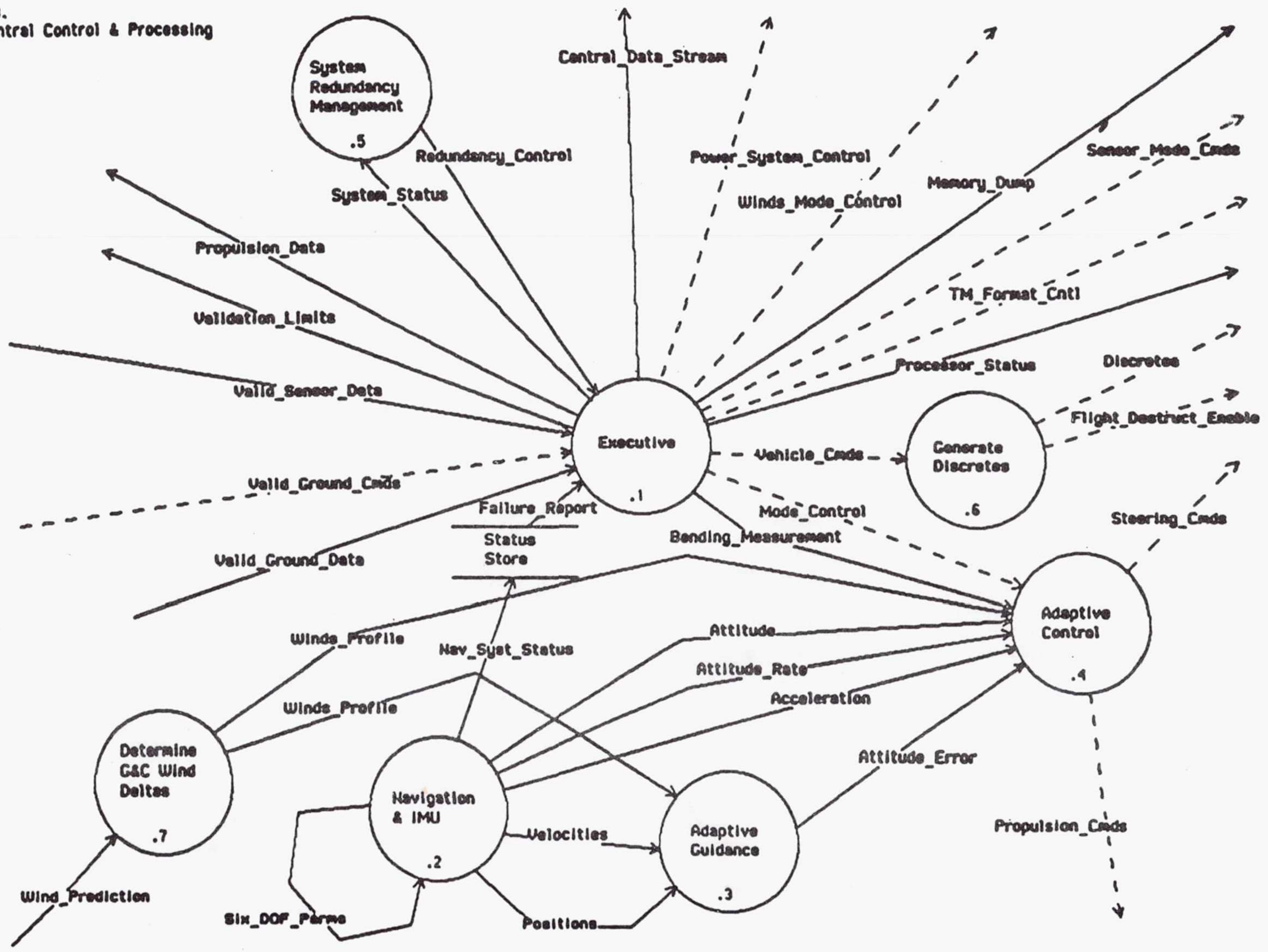
Ground_Uplink consists of Data and Commands

8;12
FLV Avionics

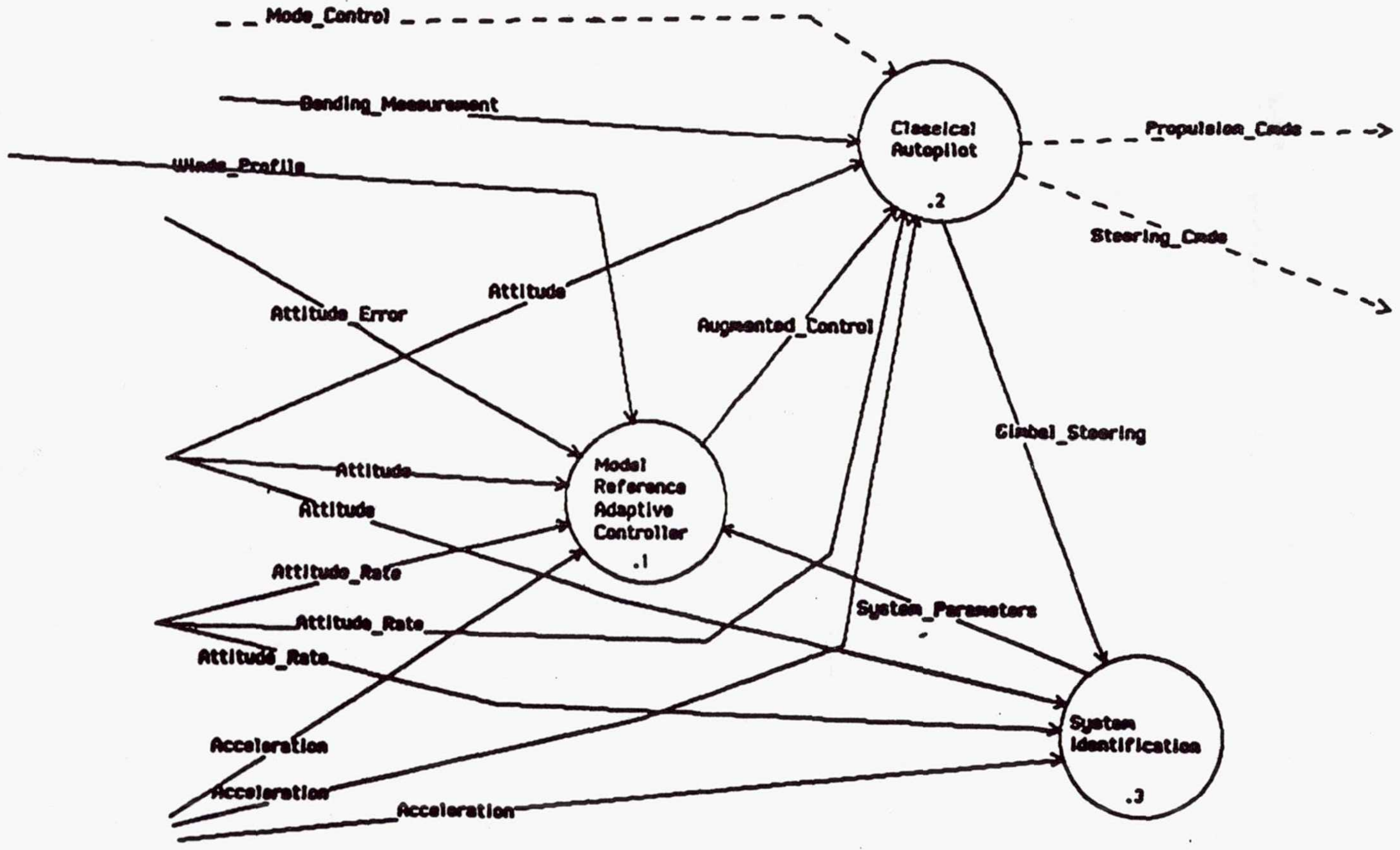


ORIGINAL PAGE IS
OF POOR QUALITY

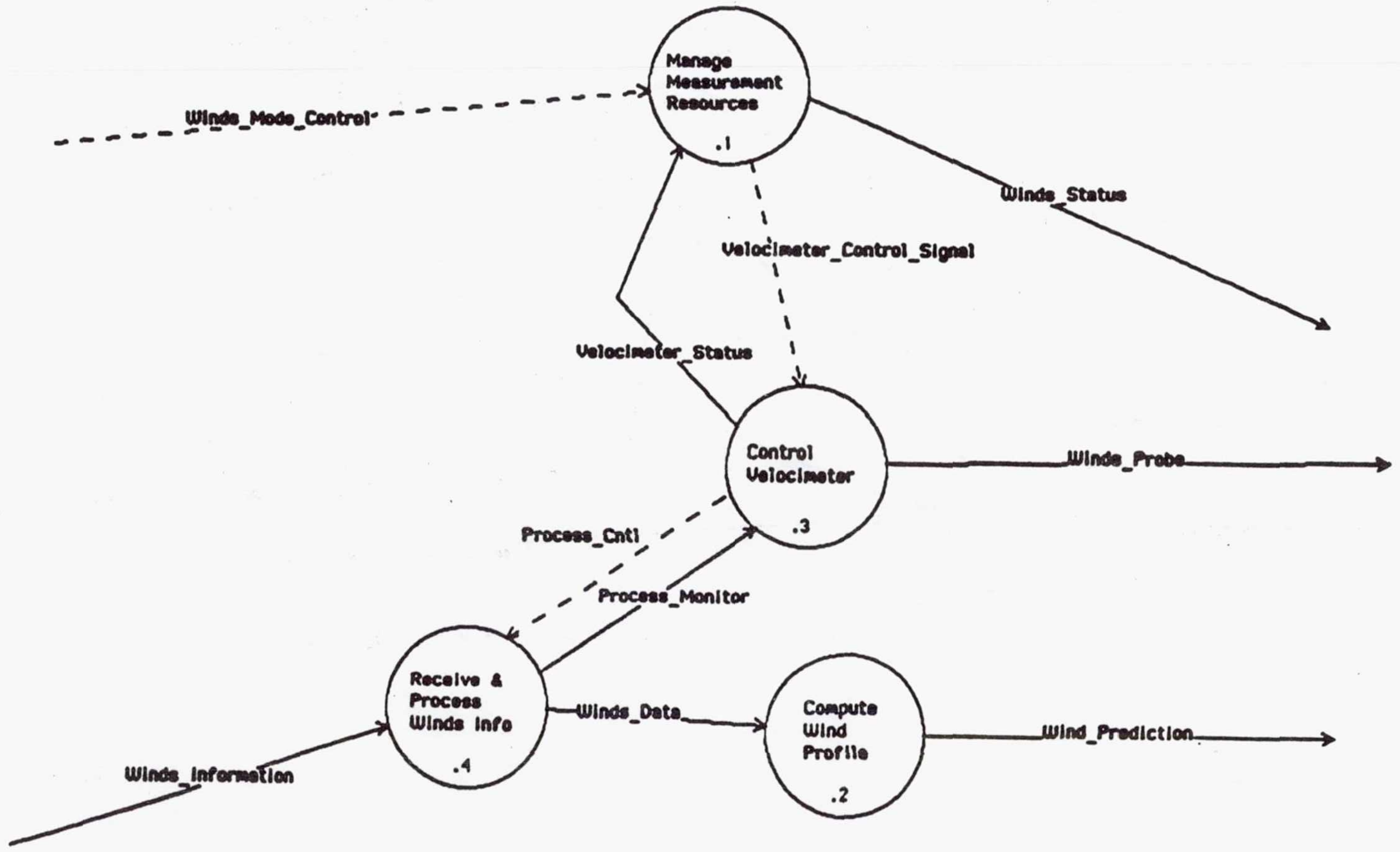
1;8.
Central Control & Processing



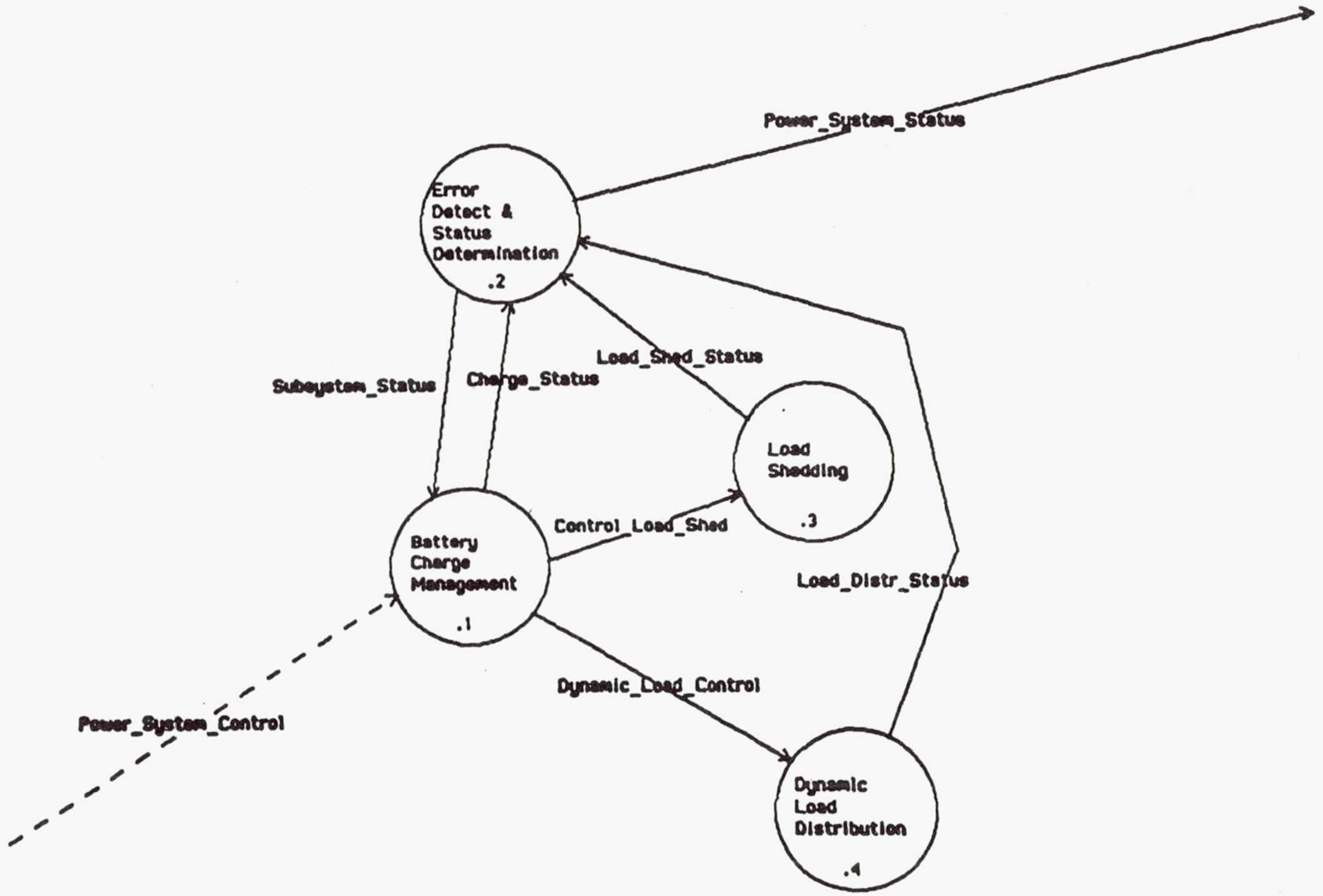
1.4;4
Adaptive Control



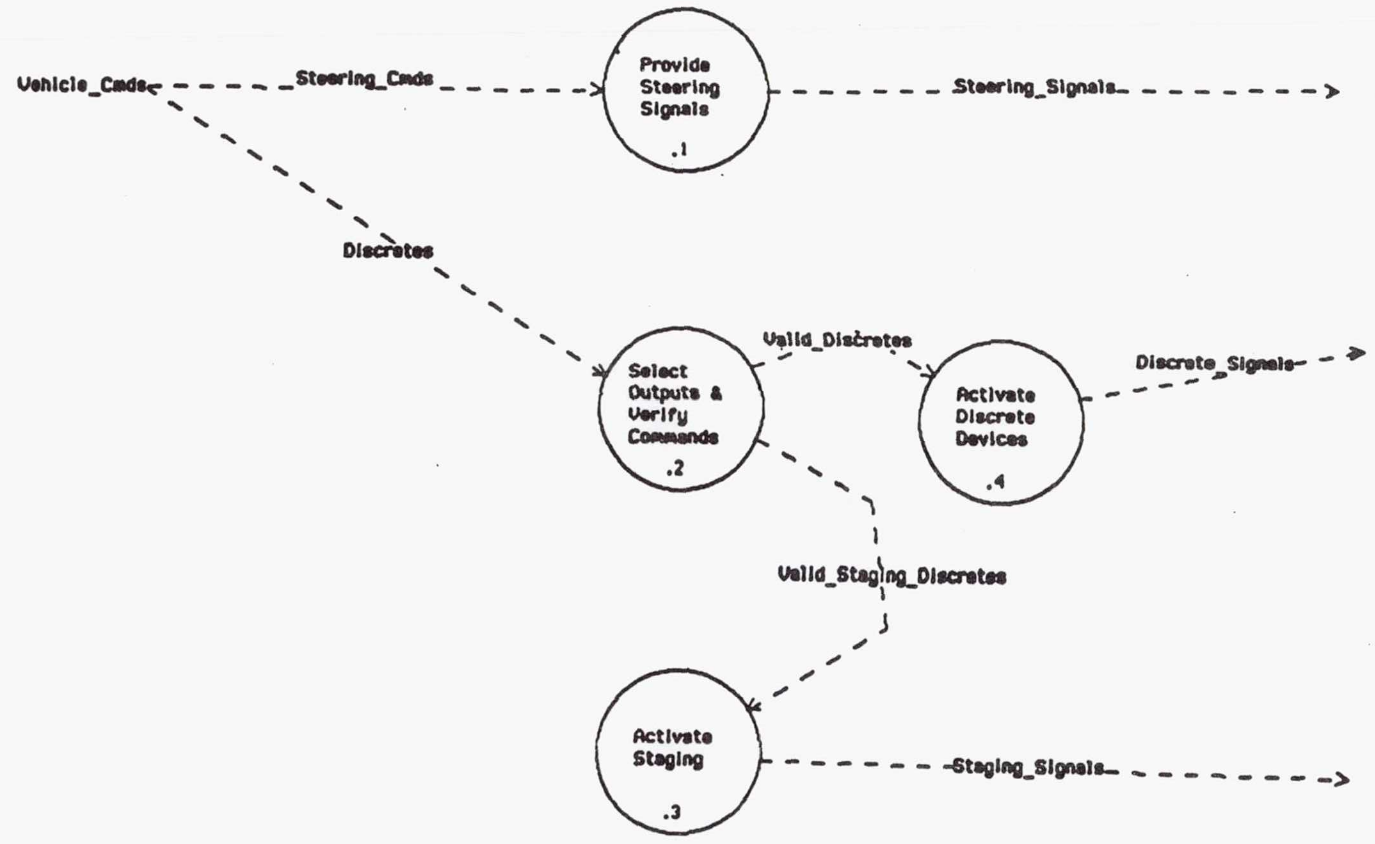
2;4
Winds Ahead Determination



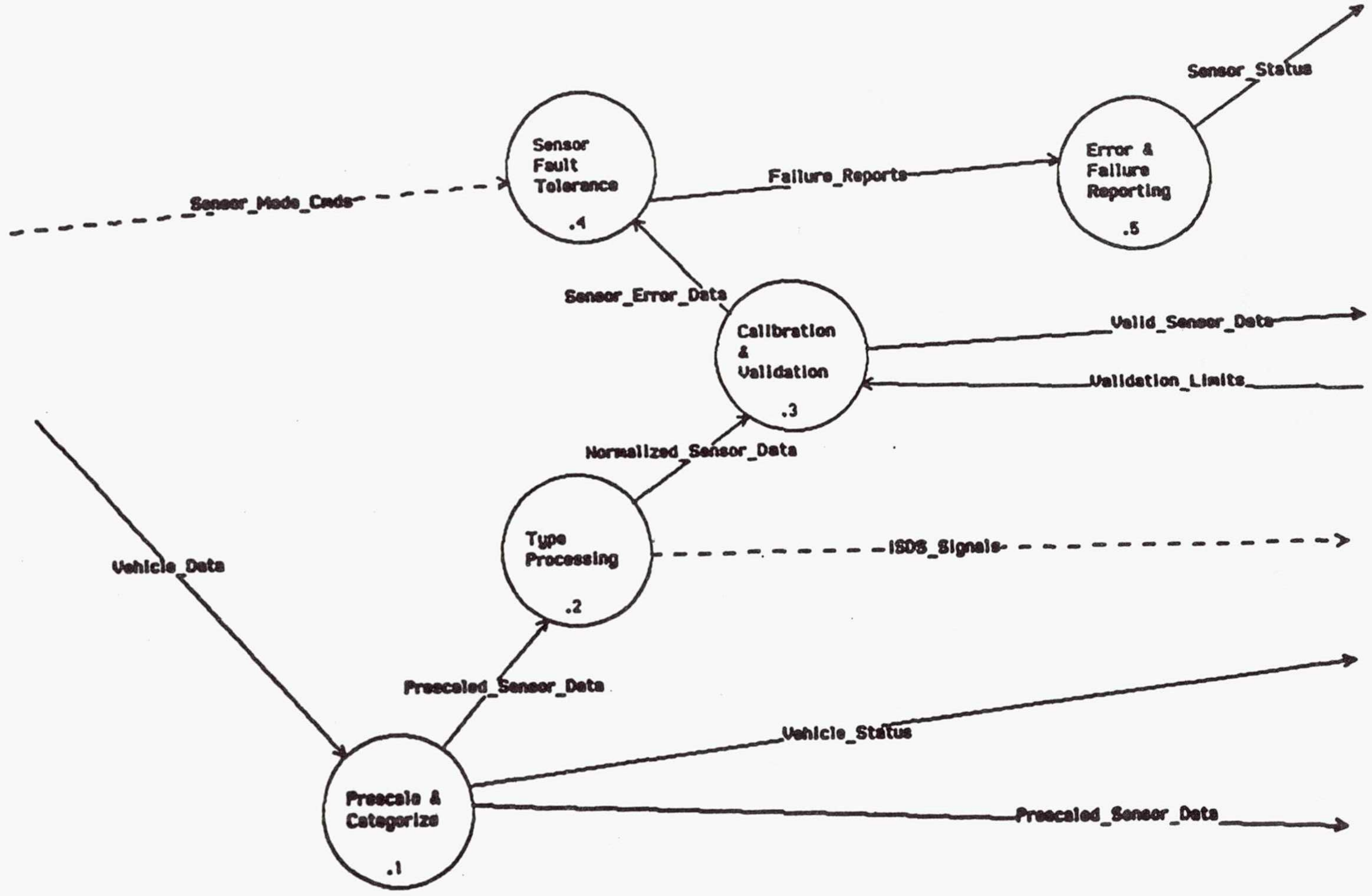
3;2
Vehicle Power System Management



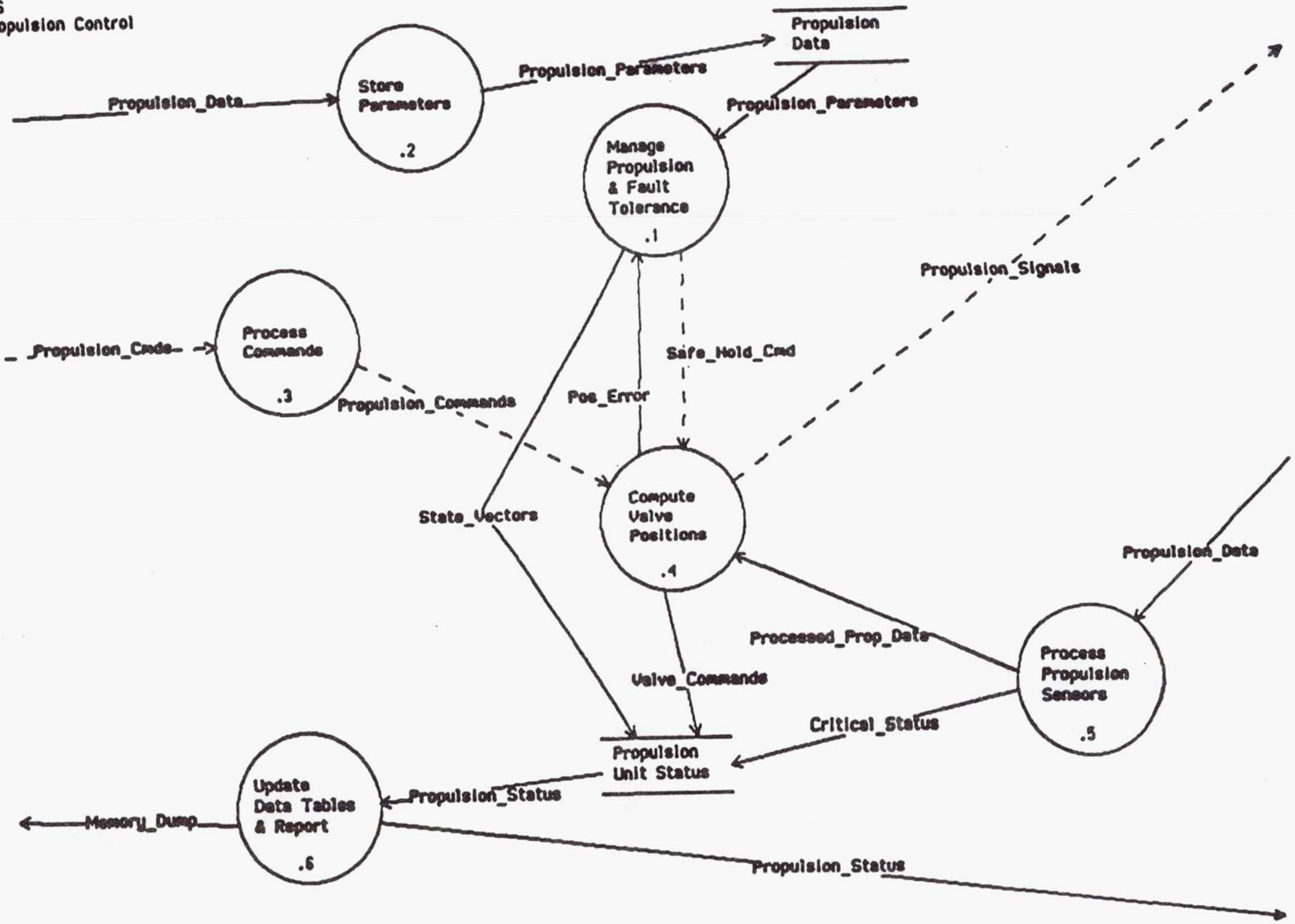
4;4
Steering & Staging Control



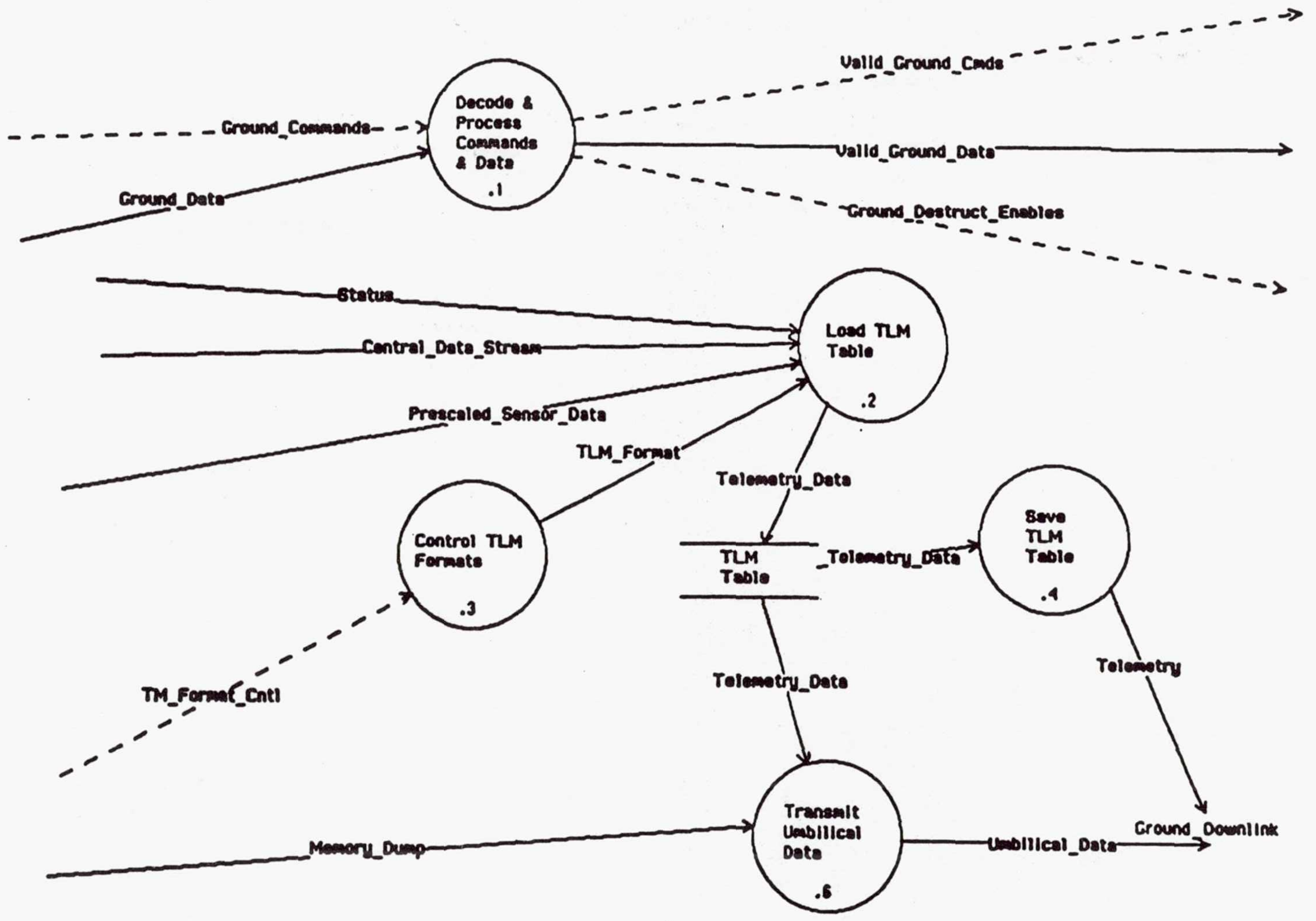
5;5
Sensor Processing



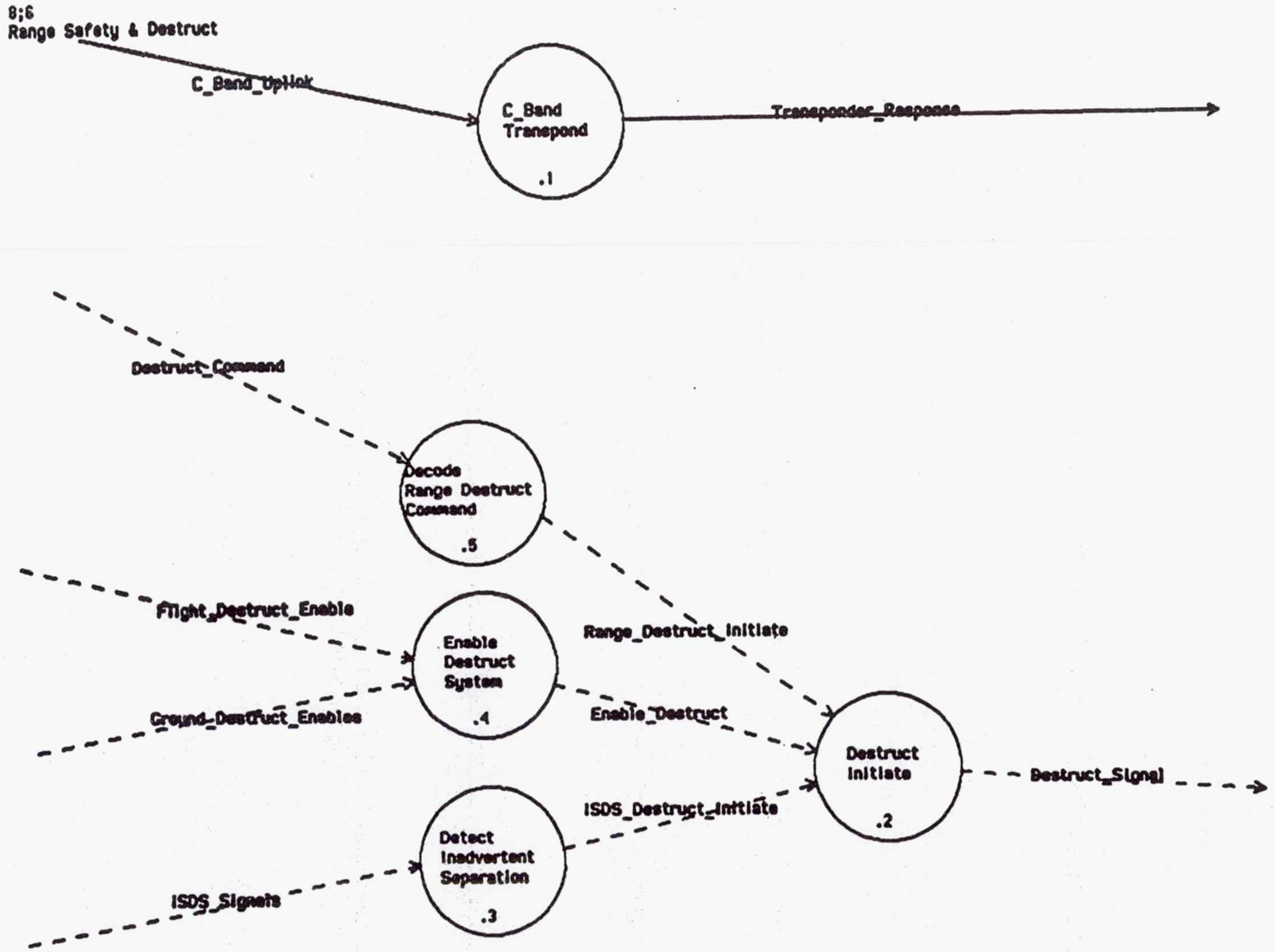
6;6
Propulsion Control



7;5
Command & Telemetry Processing



ORIGINAL PAGE IS
OF POOR QUALITY


8;6
Range Safety & Destruct
C_Band_Uplink
C_Band Transpond
.1
Transponder_Response
Destruct_Command
Decode Range Destruct Command
.5
Flight_Destruct_Enable
Enable Destruct System
.4
Ground_Destruct_Enables
Range_Destruct_Initiate
Enable_Destruct
Destruct Initiate
.2
Destruct_Signal
ISDS_Destruct_Initiate
Detect Inadvertent Separation
.3
ISDS_Signals

APPENDIX B

OS PERFORMANCE MODELING FOR DISTRIBUTED REAL-TIME SYSTEMS

Introduction

In designing and developing a highly reliable and fault-tolerant system, the use of performance modeling at various stages of the process may provide useful information to system developers. This task investigates issues on the role of performance modeling in an integrated design environment. In particular, it examines modeling abstractions and modeling fidelity issues for incorporating operating system (OS) characteristics into system performance models. Given the application of interest, the work concentrates on real-time distributed operating systems rather than general purpose systems.

Specific goals and objectives were to identify useful performance abstractions for operating systems that maintain modeling fidelity and to demonstrate their use in a simple application model.

Performance Modeling

In keeping with good engineering practice, the design process is multiply and recursively iterative. In this context, the design process proceeds from the abstract to the specific. Performance modeling at each stage and substage can help guide the decomposition process further.

Performance can mean different things at different levels of abstraction. Higher levels of abstraction provide relatively crude measures, while lower levels can provide more accurate measures, as shown by the modeling experiments described below.

The most important question for any modeling effort is clear: how can lower level details be abstracted without losing too much fidelity in the model? In other words, what can be abstracted and what cannot? To some extent, the answer is dependent upon the importance of the issues under experimental study, i.e., the dependent variables for the modeling effort, relative to the particular system.

Since performance is only one of the many design requirements a system must meet, modeling trade-offs need to be made. One such trade-off is modeling fidelity versus cost. High fidelity models may be expected to give more accurate results, but may be more costly to develop. Ideally, system modeling should allow developers to select a level of detail appropriate to their particular needs and constraints.

In the realm of performance modeling, the partitioning of the system is a very important issue. Modeling the application, the OS system services, the OS task controls, or the architecture and its interconnection network, requires different perspectives on performance issues. The performance attributes at each level of

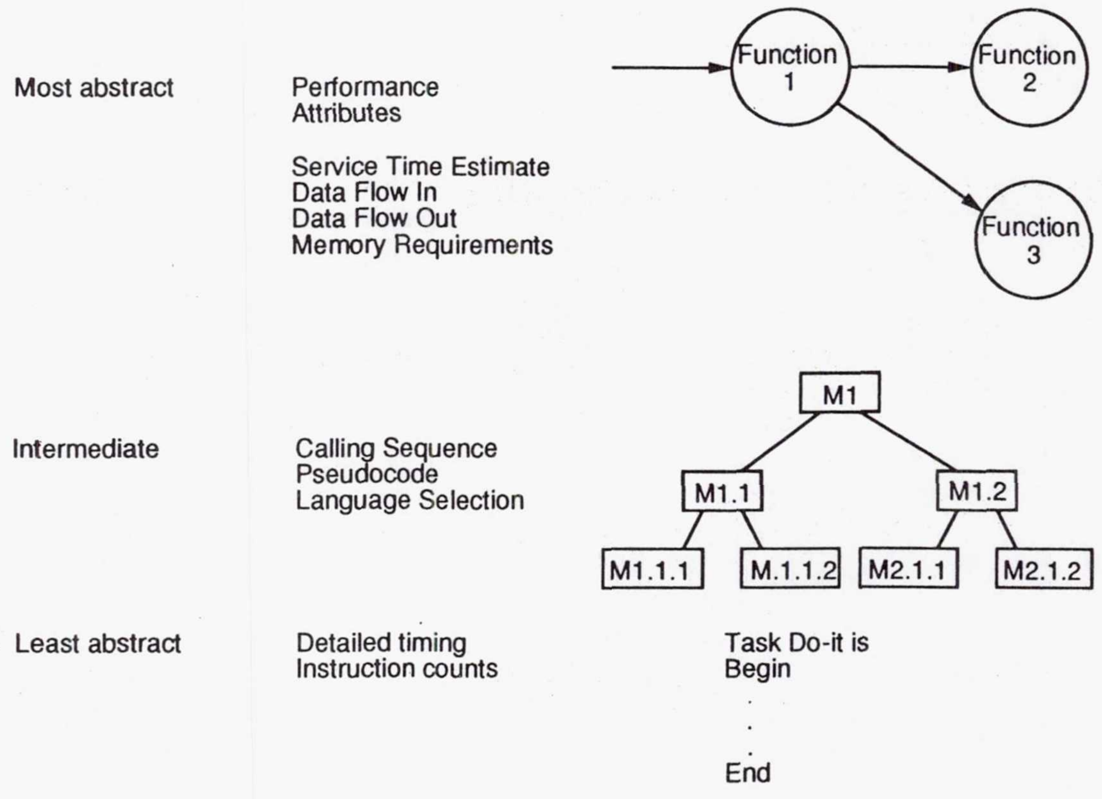


Figure 15. Application Software

modeling abstraction vary according to the system element being modeled.

Suppose that a model of operating system abstractions for task control is to be created. The operating system would decide when tasks get executed, blocked, or suspended. Performance metrics, such as processor utilization and task response times (which depend on OS scheduling policy), can be used. Some task control mechanisms are relatively simple and can be modeled using performance simulation tools like ADAS, such as fixed schedules and preemptive priority. It is also true that task control can be very complex to model, such as when modeling the Ada tasking semantics since so much is hidden by the language and its run-time system.

If application software is being modeled, the most abstract view might use an estimate of service time, examine data flow in and out, and consider memory requirements. An intermediate level of abstraction might consider calling sequences and pseudocode for specific pieces of the algorithm being modeled. The most granular and least abstracted view might model detailed timing constraints and instruction counts. Figure 15 illustrates this point.

Table 1. Operating Systems Abstractions for System Services

Level of Abstraction	Performance Attribute
Most abstract	% Overhead for task
Intermediate level	# OS calls × time per call by a task
Lower level	Dynamic model of OS services (e.g., shared memory contention)
Least abstract	Actual run time environment

If operating system services are being modeled, the most abstract view might estimate the percentage of overhead for a task service. An intermediate might evaluate performance in terms of the number of OS calls multiplied by the total amount of time per call by a task. A further degree of refinement in detail might model shared memory contention. Table 1 illustrates this point.

If interprocessor communication is being modeled, the most abstract view might model communication in terms of a fixed size delay. An intermediate level might take the model further and determine the delay based on channel bandwidth, size of data message, and overhead related to a network packet. Even more detail could be achieved by modeling the changes in delay dynamically. The least abstracted view would have detailed timings based on all of the factors that enter the transmission process. Figure 16 illustrates this point.

In summary, useful performance abstractions exist for applications, operating systems, and architecture modeling. In the case of operating system control and interprocess communication (the subjects of the experiments described below), some control and communication semantics can be modeled relatively easily. More complex task control semantics, such as with Ada tasking, are more difficult to model.

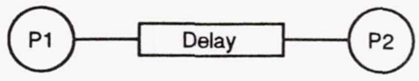
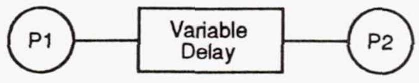
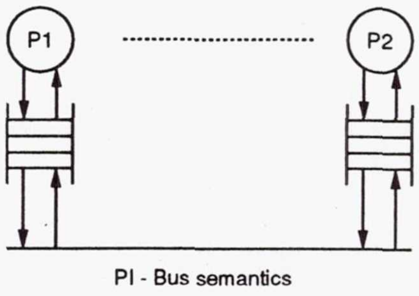
Level of Abstraction	Performance Attribute	Representation
Most abstract	Fixed Delay	
Intermediate levels	Delay is function of channel bandwidth, amount of data, and packet overhead	
	Dynamic model with channel semantics	
Least abstract	Detailed communication timings	ISA/RTL simulation models in VHDL

Figure 16. Interprocessor Communication

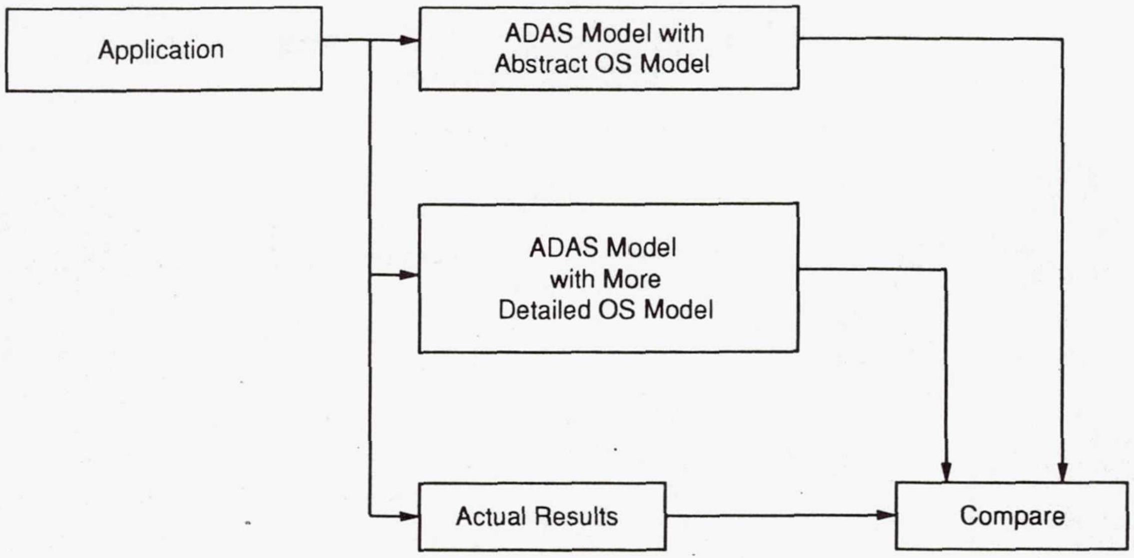

Application
ADAS Model with Abstract OS Model
ADAS Model with More Detailed OS Model
Actual Results
Compare

Figure 17. Demonstration

Experiment

This experiment illustrates the relationship of model fidelity to levels of abstraction. It shows that the addition of details to the model produces simulation results that have increasing degrees of fidelity. The experiment was constructed so that performance measures taken from models containing operating system and communication abstractions could be compared to performance measures taken from an actual prototypical implementation. The models were constructed and simulated using ADAS. They consisted of one very abstract model and a second more detailed model, which allowed for differing degrees of abstraction in functional simulation with the same topographical model. Figure 17 illustrates the experimental paradigm.

Description

The system application chosen for this experiment was a distributed client-server model. A single client module was included, with six server modules to provide generic services and a single Ethernet-like network for communication among processes. Each functional component was conceived to be executing on separate hardware processors. In simulation, the functional modules were utilizing their own processing elements (PEs), while the network modules were viewed as competing for the Ethernet transmission medium.

As mentioned above, the ADAS models included two basic levels of abstraction, a high level of abstraction and a more detailed level, with this second level being simulated in two modes of increasing detail. The models included operating system abstractions and communication abstractions.

With the ADAS performance and functional modeling tool, node firing delays (or the amount of simulated execution time a process uses) can be random numbers that are computed by user-defined routines during functional simulation. Random numbers are often required in simulations to model unpredictable events, such as arrival rates of requests for service and random service times.

The use of stochastic attributes to represent random events in the modeling process enhances fidelity. Random numbers are usually generated from distributions that attempt to model real-world behavior, such as Poisson distributions for queuing models, exponential and Weibull distributions for failure models, and uniform distributions for service time modeling. A particular distribution and its parameters are chosen for the application based on experimental observations and assumptions about the processes involved in the experiment. Simulations should then be carried out until performance measures stabilize. It is difficult to predict a priori how long a simulation should run; usually the experimenter should decide based on modeling experience and knowledge of the particular modeling tool.

This experiment examined average network throughput, client utilization of its processor during simulation run, and server utilizations of their separate processors during simulation run. In this context, throughput was considered to be the total amount of data that passed out of the network nodes in a given second of simulation time. Client and server utilizations were considered to be measures of the amount of simulation time, as a percentage of the total simulation time, that the individual hardware PEs were busy.

Independent Variables

This experiment used three independent variables. They were the client execution time, the server execution time, and the size of the user message. The client execution time was based on measured values from an actual prototype implementation of the application. Although it turned out to be a very small amount of time, 500 microseconds, a greater time would not have affected the essential problem being modeled. It was statically set, and remained the same, for all simulation runs.

The server execution times were chosen to be uniformly distributed about a mean value. The mean value was statically set, prior to each simulation run, with the uniform distribution being calculated dynamically for each server node firing. The mean times varied from 500 microseconds to 1 second.

User messages were varied from 64 bytes to 32 kilobytes, with the specific size statically set prior to each set of simulation runs (ten settings). Message size was incorporated directly into the network simulation models as network node firing delays in the first model, and as production and consumption of simulation tokens in the second model.

Sets of simulation runs were carried out, with each of the ten user message sizes being held constant for all of the twelve server mean values, resulting in 120 data points.

Model One

Model One was the most abstract and coarsely-grained of the experimental models. It assumed that the execution times for both the client and server models encompassed the application functions and system networking functions, with additional network functions modeled as part of the network nodes. Again, firing delays for the node execution times were taken from an actual prototype application implementation.

Table 2 shows the message cycle times as actually measured in a prototype implementation. The times shown are the average times taken for a message to be read, processed, and for anew message to be sent. Since process time for the application was known to be very small, the largest portion of the times represents message overhead for reading and writing.

Table 3 shows the firing delays used in Model 1 to simulate the passing of messages, and application code execution times. The server delay was incremented by a reading and writing cost constant for each message size. The network delay was estimated to be a fraction of the message size (in bits).

This model assumed that transmission time across the network was a function of network bandwidth and the length of a user message. The network node firing delay was set to be a function of the size of the user message as a result of this assumption. The total user message was assumed to be transmitted and received without being broken down into network transmission packets. Coarse-grained contention for network services was modeled in terms of the network nodes competing for the network hardware resource. Figure 18 illustrates the ADAS software graph of Model One.

Message Size (bytes)	Delay (msec)
64	6.737
128	6.791
256	7.023
512	7.533
1024	8.870
2048	12.258
4096	17.619
8192	26.961
16384	53.356
32768	97.364

Table 2. Measured Delays for Message Cycle

Message Size (bytes)	Client Delay (msec)	Server Delay (msec)	Net Delay (msec)
64	3677	3678 + random number	0.0512
128	3713	3713 + random number	0.1024
256	3868	3868 + random number	0.2048
512	4208	4208 + random number	0.4096
1024	5099	5099 + random number	0.8192
2048	6544	6544 + random number	1.6384
4096	9304	9304 + random number	3.2768
8192	13090	13090 + random number	6.5536
16384	32469	32469 + random number	13.1072
32768	46187	46187 + random number	26.2144

Table 3. Component Delays for Model 1

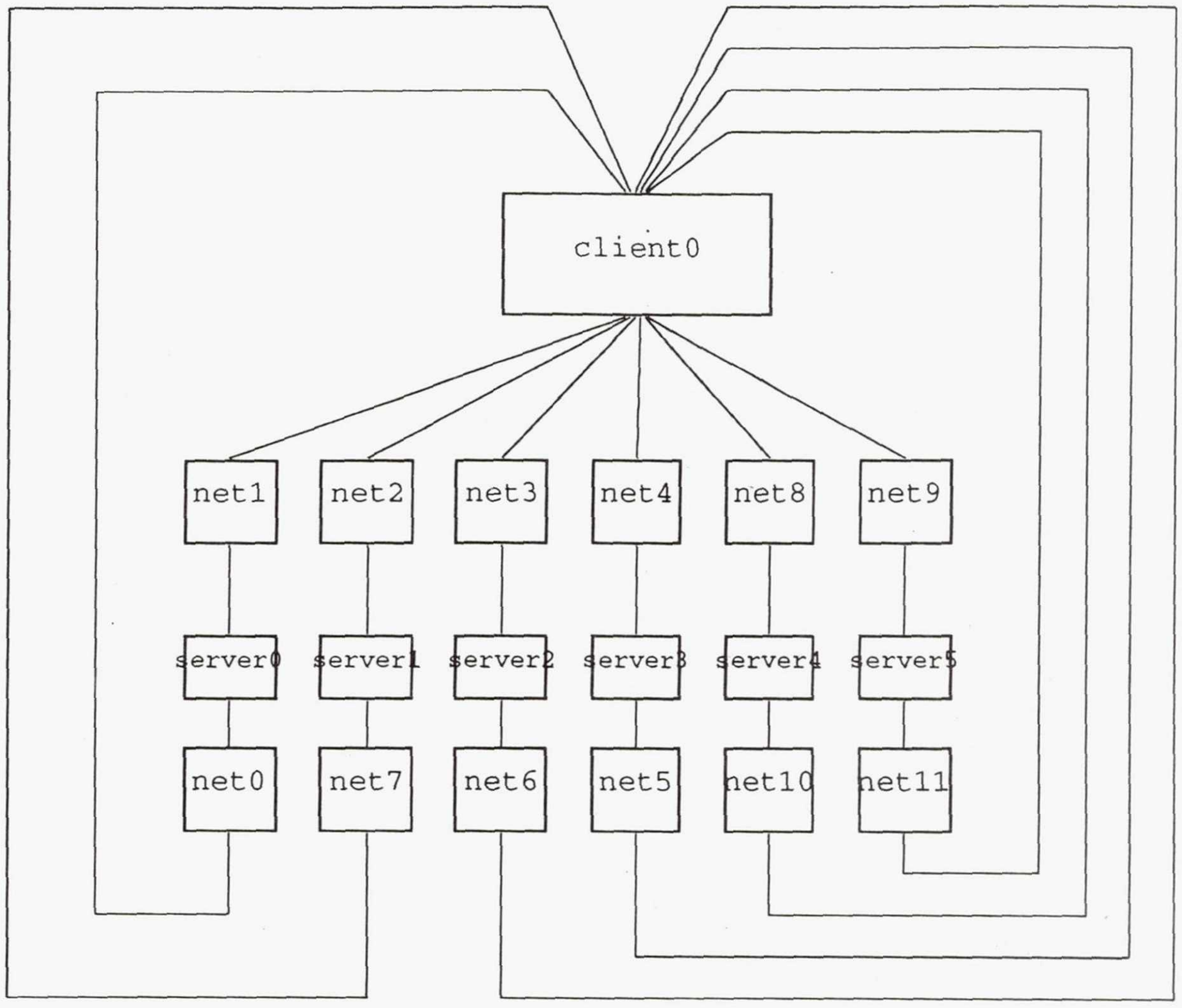


Figure 18. Model One

Model Two

Model Two represented a less abstract model with the introduction of two finer levels of detail. In these two increasingly detailed views of the system, the client and server modules were decomposed into application functions and I/O functions. This more closely represents the way that the operating system and its network protocol software actually function in relationship to any specific application. The ADAS graph topography was hierarchical in its decomposition of the model, and it was the same for both versions of Model Two. Figures 19, 20, 21, and 22 illustrate the ADAS software graph hierarchy for the Model Two topography.

Model Two — First Version

The first version of Model Two made a similar assumption about user message transmission as Model One. In both cases the total user message was transmitted and received as a unit. However, in this case, the transmission time was a function of the number of Ethernet packets required for a particular message size (packet sizes ranging from 64 bytes to 1024 bytes, even though Ethernet packets can range up to 1518 bytes). Based on the prototype implementation, the amount of Ethernet transmission time was calculated (exclusive of operating system network services) for messages of different sizes. The firing delay for the network nodes was statically calculated in terms of this transmission time, and the simulations were run using this calculation.

Table 4 shows the component delays for Model 2 Version 1. The client and sewer nodes have been decomposed to show the actual reading and writing of messages. The times again were based on the results obtained from the prototype implementation for each message size. The next delay was based message size relative to maximum network bandwidth.

This version represented a more detailed and accurate view of data transmission in the system. Message transmission interleaving was again modeled in terms of contention for the Ethernet hardware resource. The next version refined this model further.

Model Two — Second Version

Using the same graph hierarchy as the first version of Model Two, the second version altered the network model assumptions. In this case, the user messages were decomposed into Ethernet packets for transmission, each packet no larger than 1024 bytes (arbitrarily chosen for modeling convenience). A single token was used to

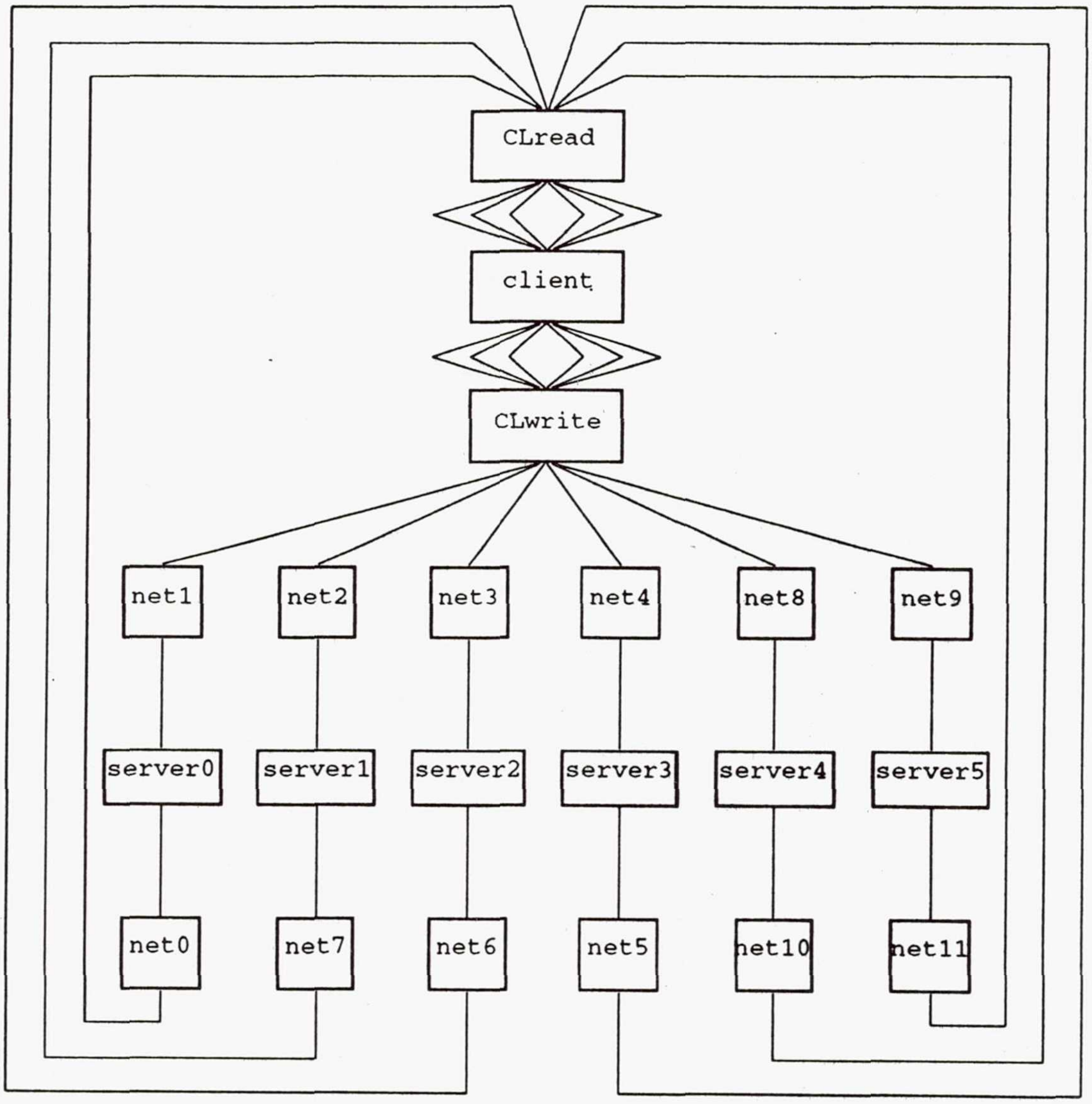


Figure 19. Model Two

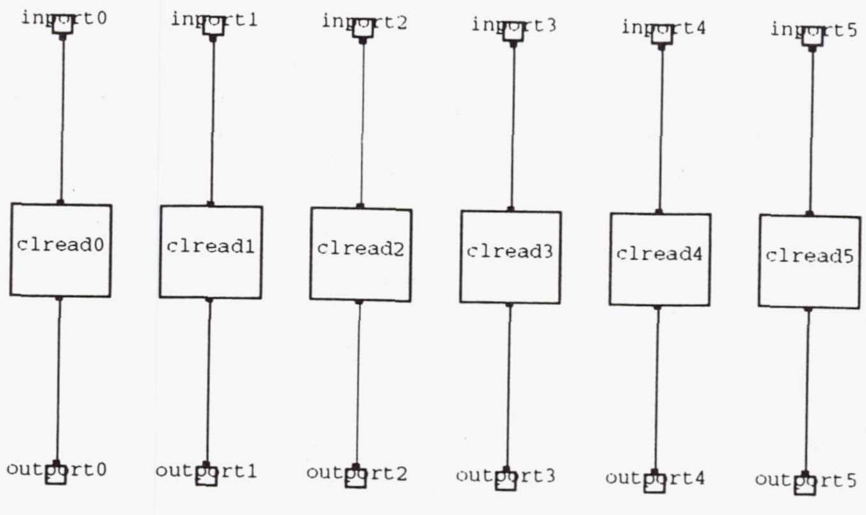


Figure 20. CLREAD.SWG

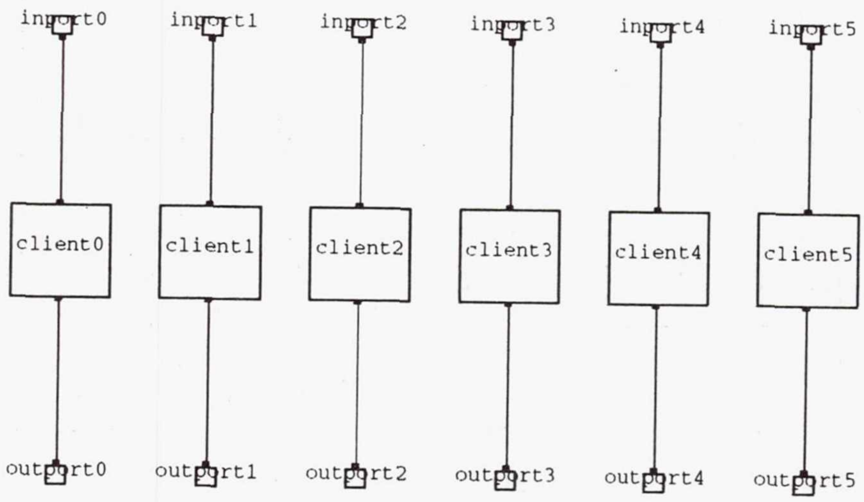


Figure 21. CLIENT.SWG

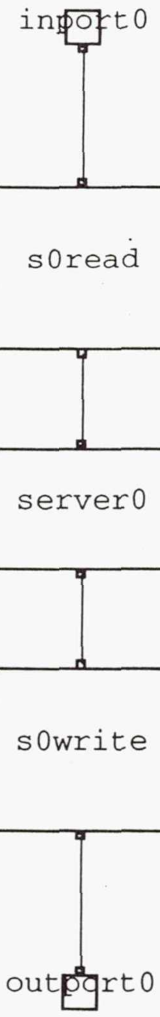


Figure 22. Server 0

Message Size (bytes)	Client Read (msec)	Client Write (msec)	Server Read (msec)	Server Write (msec)	Net (msec)
64	1672	1672	1672	1672	58
128	1690	1690	1690	1690	109
256	1767	1767	1767	1767	211
512	1937	1937	1937	1937	416
1024	2383	2383	2383	2383	826
2048	3105	3105	3105	3105	1652
4096	4485	4485	4485	4485	3304
8192	6378	6378	6378	6378	6608
16384	16068	16068	16068	16068	132
32768	22927	22927	22927	22927	26432

Table 4. Component Delays for Model 2 Version 1

model a single packet of user message data. (*Note:* Recall that in Model One and the first version of Model Two the entire user message was sent and received as a single token.) The total user message thus consisted of the number of token packets required to send the message by the actual implementation. Consumption and production of tokens from and to network nodes were scaled to the size of the user message (which was statically set for the entire simulation run). The client-read nodes and the server-read nodes consumed the number of tokens representing a full user message. The client-write node and the server-write nodes produced the number of tokens representing a full user message. The network nodes could only consume and produce a single token. Contention for the network hardware under these conditions simulated the interleaving of packets and the consequences of message delays in transmission for the application software in an actual application. Each network node execution time (firing delay) was based on a single packet's transmission costs.

With the modification of the produces and consumes relative to message transmission and the single token as single packet modeling assumption, this version of Model Two further refines the application-OS-network interactions.

Table 5 show the most granular model, Model 2 Version 2. The client and sewer nodes were again decomposed into reading and writing components with Airing delay times taken from the prototype implementation. The net delays are also the same as in Version 1, up to the 2048 bytes message size. In Version 2 tokens represent packets of data. The model uses a maximum packet size of 1024 bytes. Larger messages require more packets. The net node is modeled to only transfer a packet at a time causing larger messages to be transmitted piece by piece, interleaved with packets containing other messages. This approach is of higher fidelity to the way that network transmission actually works.

Message Size (bytes)	Client Read (msec)	Tokens Consumed	Client Write (msec)	Tokens Produced	Server Read (msec)	Tokens Consumed	Server Write (msec)	Tokens Produced	Net Delay (msec)
64	1672	1	1672	1	1672	1	1672	1	58
128	1690	1	1690	1	1690	1	1690	1	109
256	1767	1	1767	1	1767	1	1767	1	211
512	1937	1	1937	1	1937	1	1937	1	416
1024	2383	1	2383	1	2383	1	2383	1	826
2048	3105	2	3105	2	3105	2	3105	2	826
4096	4485	4	4485	4	4485	4	4485	4	826
8192	6378	8	6378	8	6378	8	6378	8	826
16384	16068	16	16068	16	16068	16	16068	16	826
32468	22927	32	22927	32	22927	32	22927	32	826

Table 5. Component Delays for Model 2 Version 2

Prototype Testbed

The prototype testbed consisted of a VAXstation II/GPX with Ethernet interface and twenty-two rtVAX 1000 systems, also with Ethernet interfaces. As in the ADAS models, the software client and servers each ran on a separate rtVAX 1000. The testbed used the VAXELN real-time operating environment. Figure 23 illustrates the hardware testbed.

The software design for prototype application was similar to the ADAS graph models in concept. Task PEs, or servers, received input messages and performed a synthetic computational task with executive times set as in the ADAS models. Total time for each task was accumulated with high precision by the server process, and finally an output message was transmitted.

A LoopDriver PE, or client, caused Task PEs (servers) to repeat their operations for many iterations. The start-to-finish time for Task PEs was measured based on

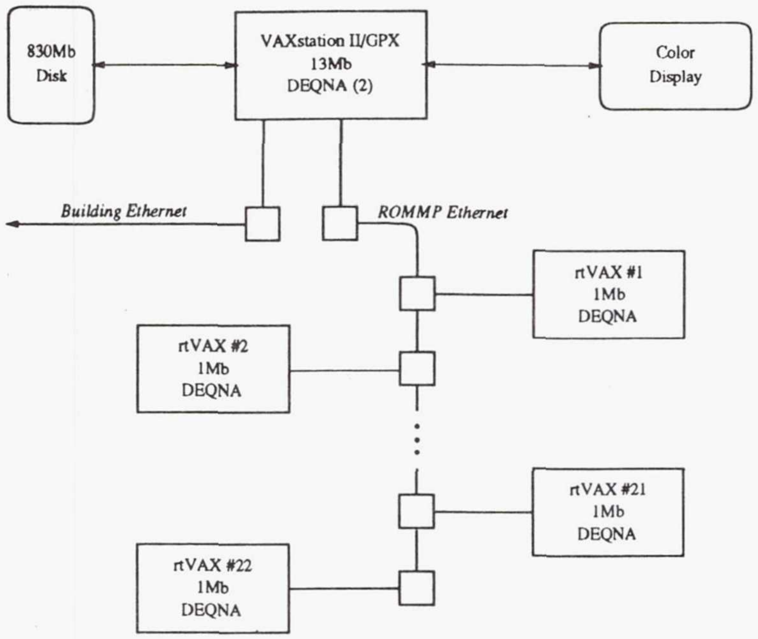


Figure 23. Hardware Configuration

recording the start-time before the first message was transmitted to each PE, and recording an end-time after receipt of last return message. The difference between the start-to-finish time for each task and the total synthetic compute time was considered to be "everything else," including I/O driver time, actual Ethernet transmit time, OS kernel time, and idle times.

A LANalyzer was used to monitor the Ethernet network, measuring the total number of packets transmitted, the number of bytes sent, and the average utilization of the network. Capture and time-stamping of the packet data allowed packet transmit times to be measured with high precision. As in the ADAS models, message size was fixed for each simulation run. Figure 24 illustrates the software paradigm for the testbed application. In this experiment, only one LoopDriver and six Tasks were used, but expansion is possible with this testbed.

Experiment Results

This experiment was intended to demonstrate that increasing refinement and detail in a model results in higher fidelity models as shown by the performance measures. The results are supportive of this assertion. While ten user message sizes from 64 bytes to 32 kilobytes were used in the experiment, three representative samples

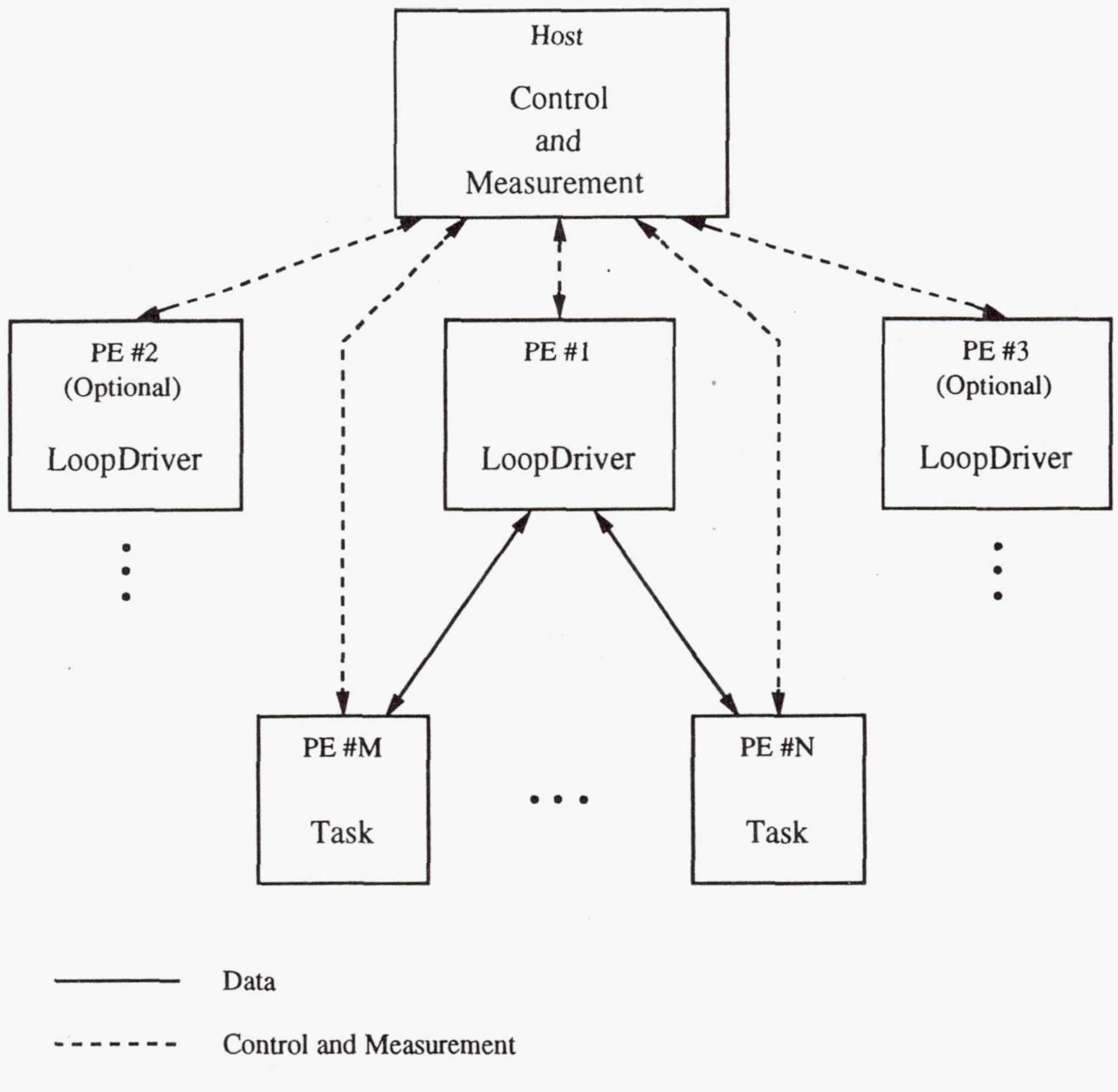


Figure 24. Software Design

of the results are presented below, representing a small user message size (64 bytes), medium user message size (2 kilobytes), and large user message size (16 kilobytes).

The dependent variables studied by this experiment were average throughput, server utilization, relative error of server utilization, network utilization, and the relative error of the network utilization.

Throughput was taken to be the total number of accesses to the network nodes during a simulation run, scaled to kilobytes per second.

Server and network utilizations were calculated to be the percentage of simulation time that the hardware resources for each were active. Relative error for each was calculated to be the difference between the ADAS utilization values and the prototype testbed values.

Since sets of simulation runs were made for each user message size, each of the dependent values was plotted against the mean server time independent variable.

In the plot figures that follow, curve A represents Model One, curve B represents the first version of Model Two, curve C represents the second version of Model Two, and curve D represents the actual implementation, where

- Model One was the simplest, lumping the application and operating system together
- the first version of Model Two refined Model One by separating the I/O functions from the application
- the second version of Model Two further refined the modeling to consider the effects of Ethernet packetizing of application data

Throughput Results

Figures 25 through 30 show the throughput results for the three data message sizes. In all cases the basic shapes of the curves show that as modeling fidelity increases, more accurate simulation results are obtained. Curve C is closer to the actual implementation than the other models. Over the range of the mean server time the curves are quite different while the network transmission time dominates the server time. As the server time increases, it begins to dominate the transmission time and the curves all converge. In the case of the 64-byte message size the model overestimates the throughput while mean server times are small, and it underestimates throughput as the server times grow. Relative error is fairly high with the smaller server times. This suggests that the model needs to be refined further to improve its fidelity. The most likely area for this refinement to be carried out is in the network services related to developing data packets for transmission. This is likely to be true for all of the results that follow.

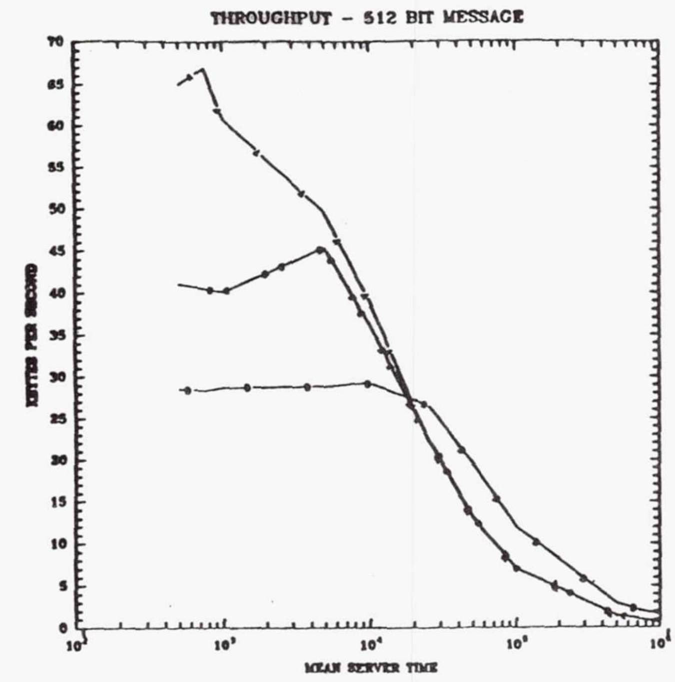


Figure 25.

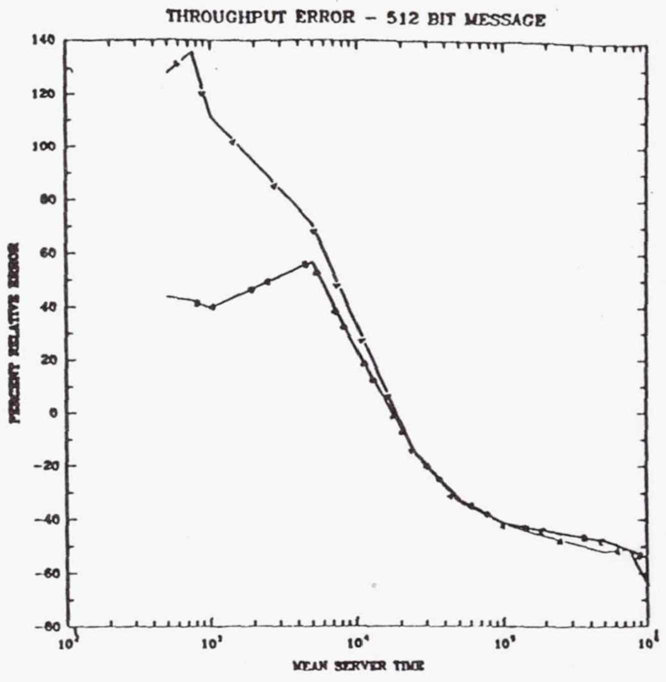


Figure 26.

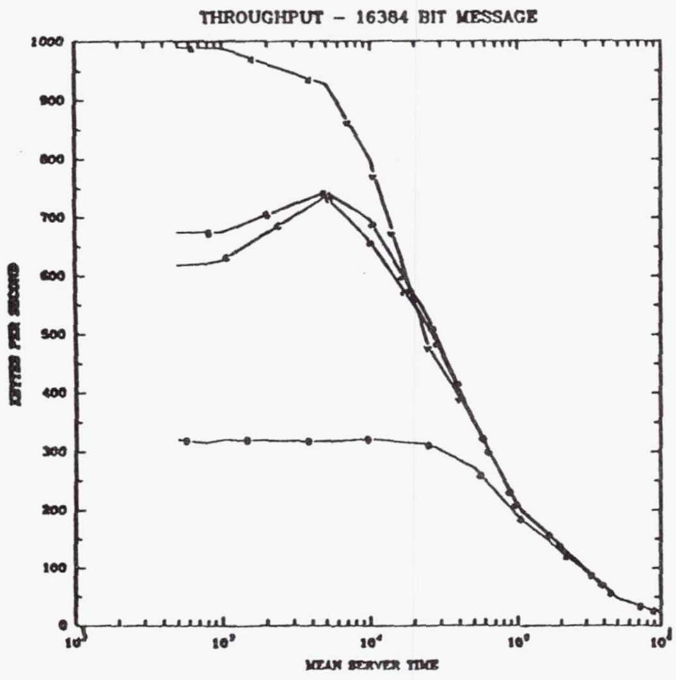


Figure 27.

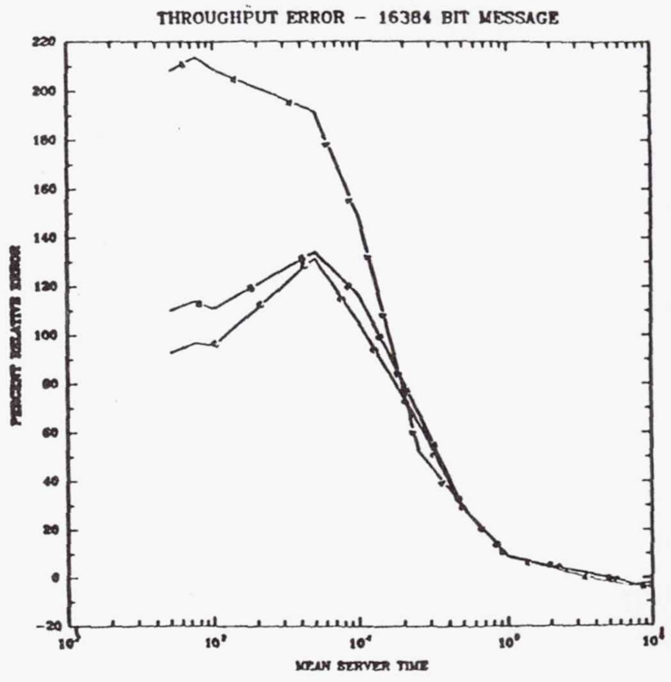


Figure 28.

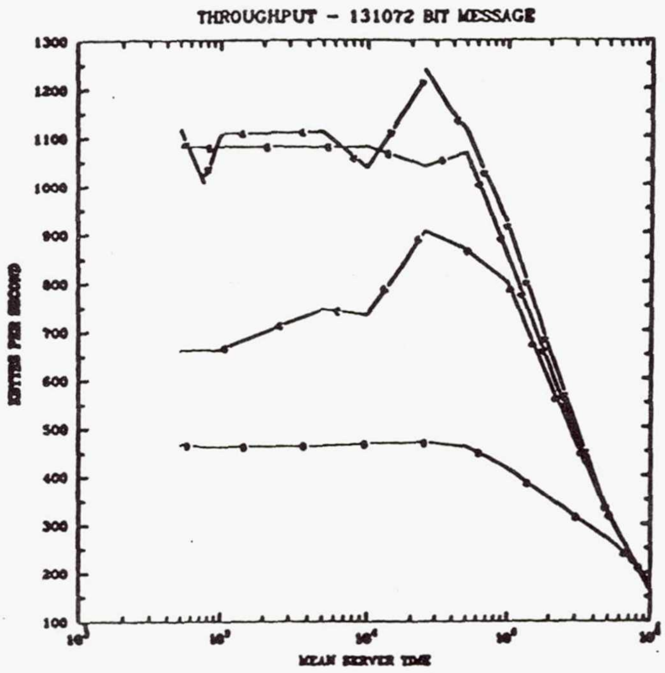


Figure 29.

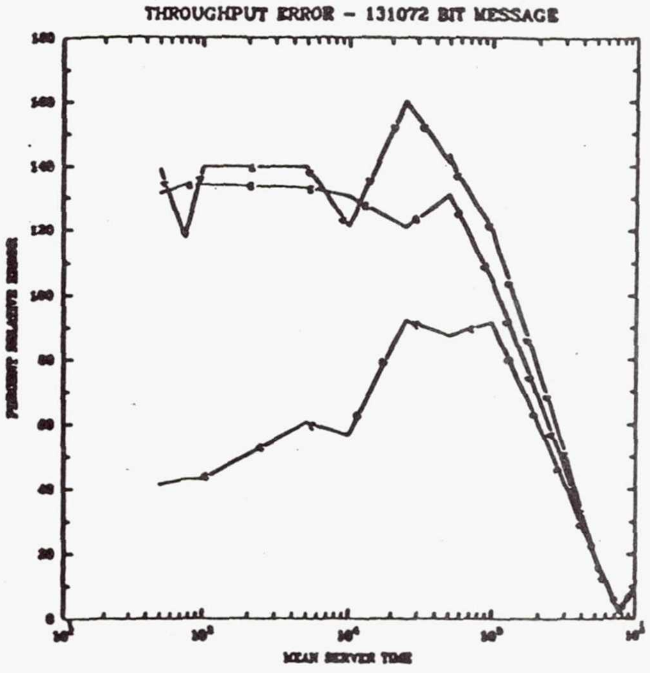


Figure 30.

C-3

Server Utilization Results

Figures 31 through 36 show the server utilization results for the three data message sizes. As with the throughput results, the two versions of Model Two more closely approximate the results from the actual implementation in terms of the shape of the curves. However, when the relative error curves are examined, even the more refined models show a high degree of relative error. Again, the increasing detail of the models show increasing fidelity, but additional iterations to refine the model further might be desirable since the models consistently overestimate the server utilization.

Network Utilization Results

Figures 37 through 42 show the network utilization results for the three data message sizes. These results are consistent with the previous sets. For small mean server times, the models overestimate the network utilization. They then underestimate it for larger server times. These results also show that the most abstract model is the least accurate. Relative error remains higher than might be desirable, suggesting that further refinement be carried out on the model.

Experiment Conclusions

This experiment indicates that the hierarchical refinement approach is a useful way to carry out a modeling effort. It suggests that any modeling effort should strive to achieve the simplest model that will give acceptable results. Clearly there will be trade-offs in cost and effort for a level considered acceptable by the modelers in any specific case.

It is also clear that any model must be validated for it to be useful. Validation implies that the model must be compared against reality to establish a range of valid results. This helps to identify problems in the model or in the measurements that were used with the model. Accurate timing information is essential to a good simulation model and this often requires specialized equipment. A model is only as good as the information it uses to generate results.

Modeling efforts are most useful when the simulation results can be reused. In other words, the effort should seek to develop a reusable model that can become a library element after validation. This lowers the cost of any modeling work over time.

Experience and judgment are important in the modeling and analysis process. Any modeling tool is only as good as the information it has available to it, and that information is a product of the professionals involved.

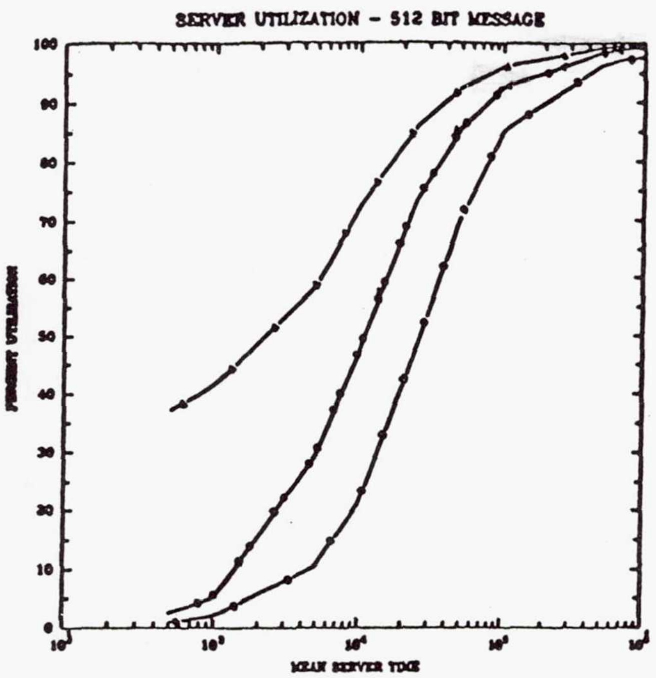


Figure 31.

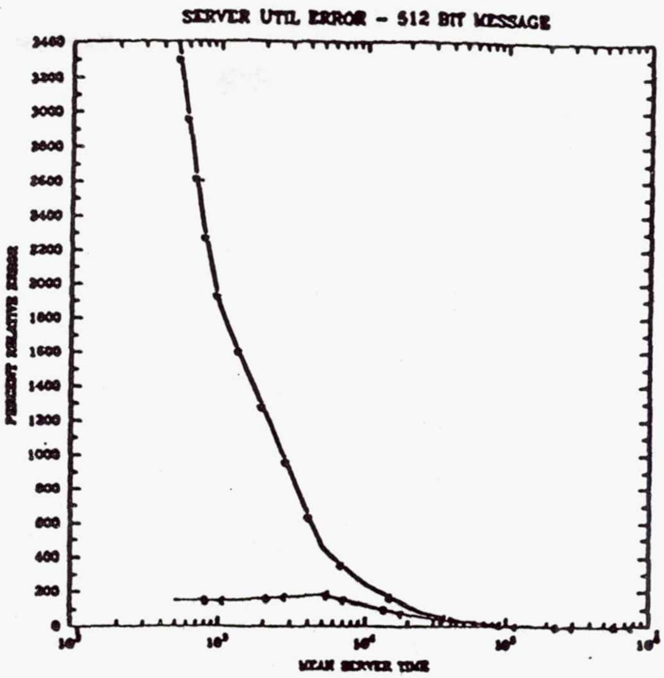


Figure 32.

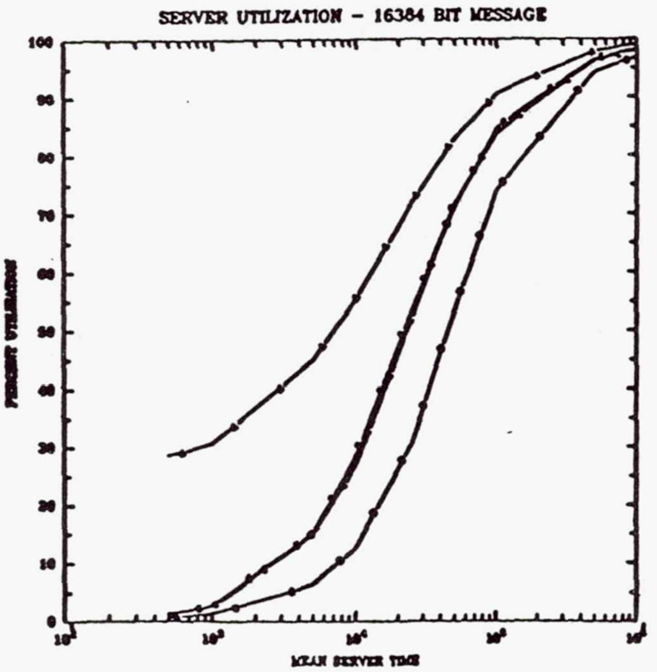


Figure 33.

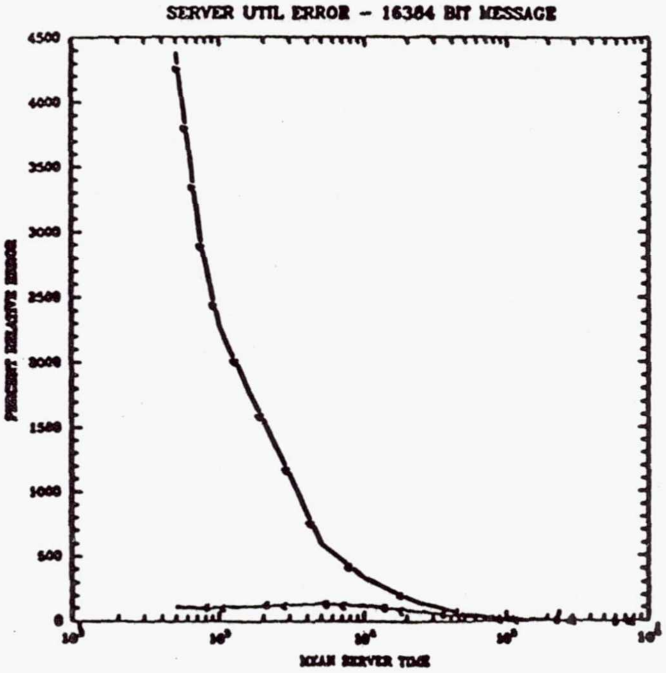


Figure 34.

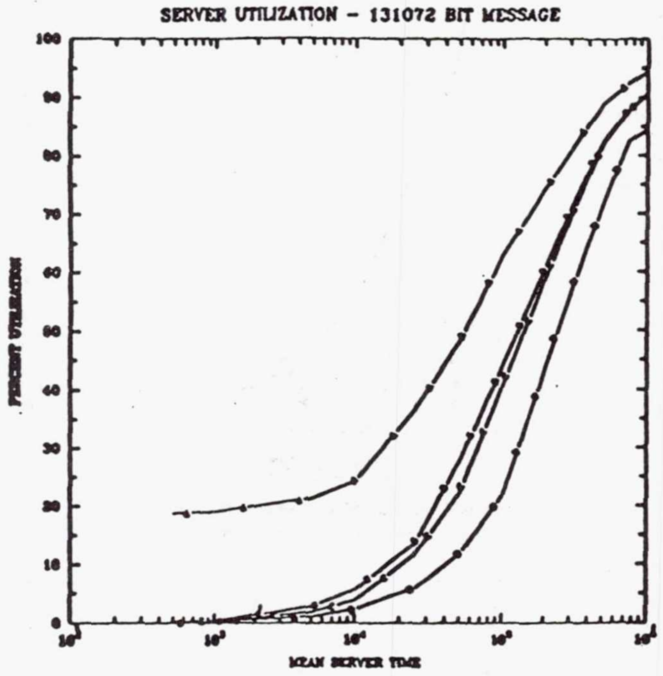


Figure 35.

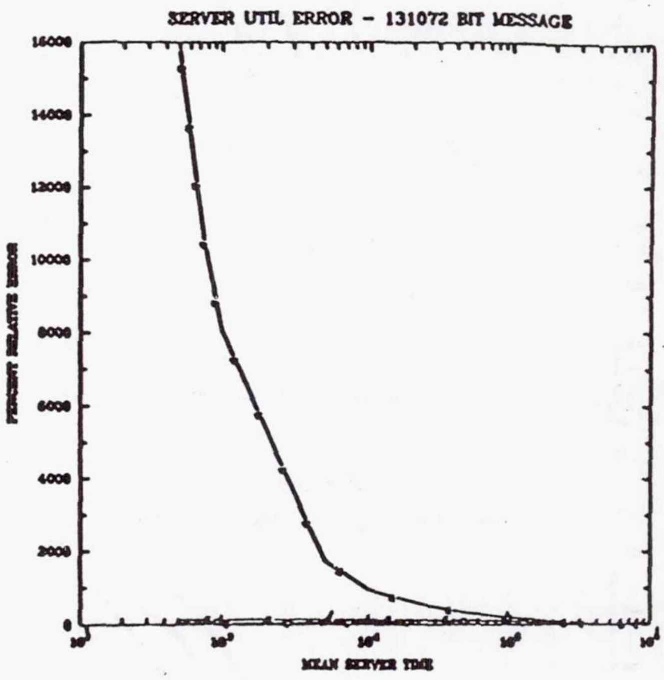


Figure 36.

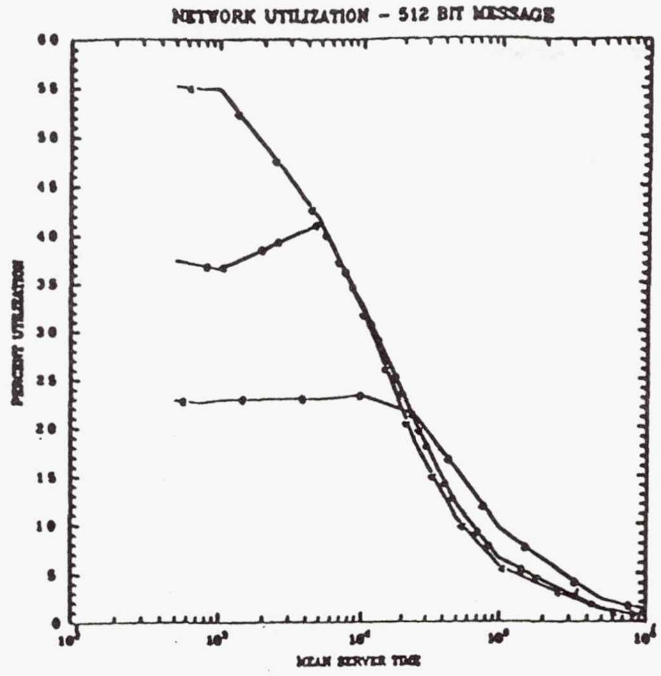


Figure 37.

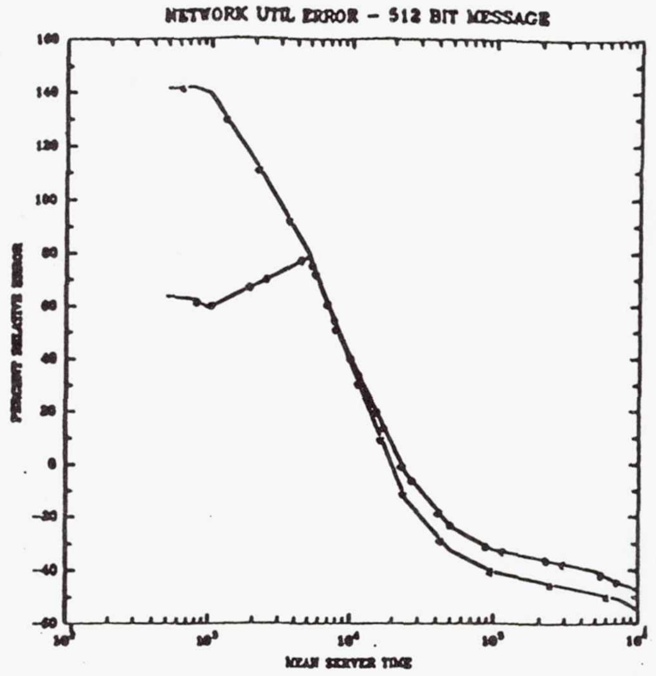


Figure 38.

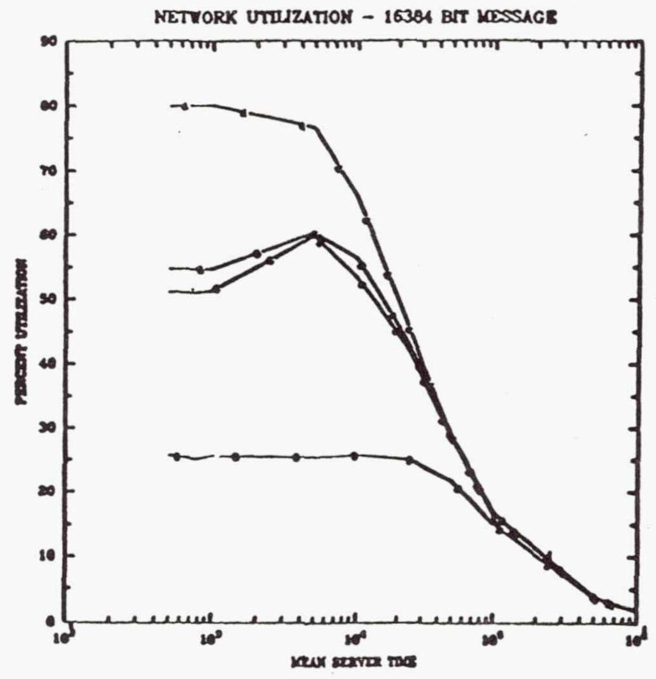


Figure 39.

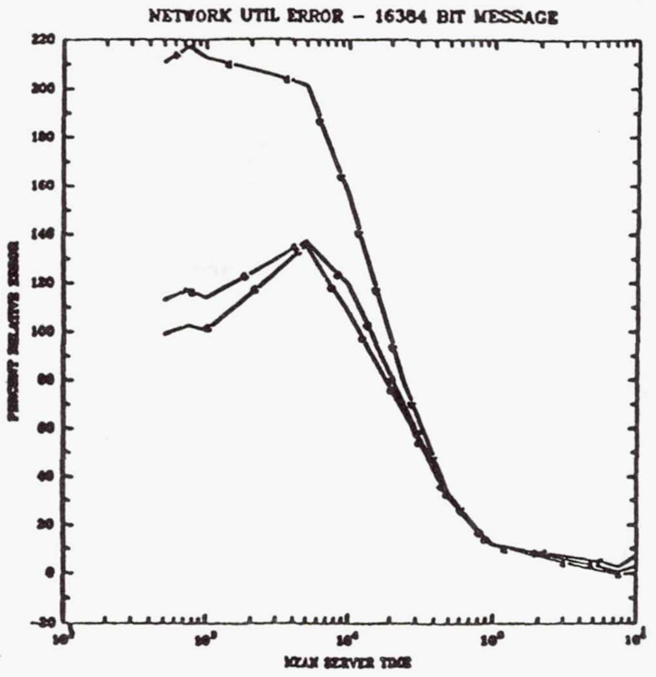


Figure 40.

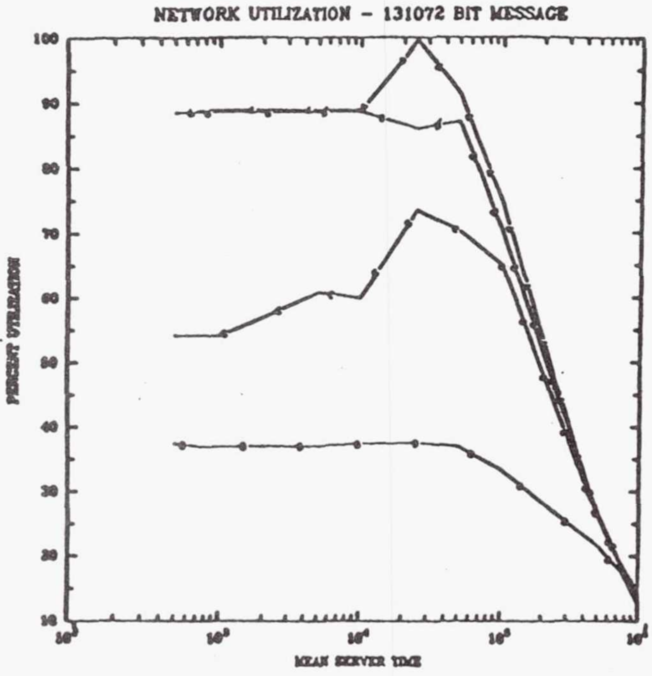


Figure 41.

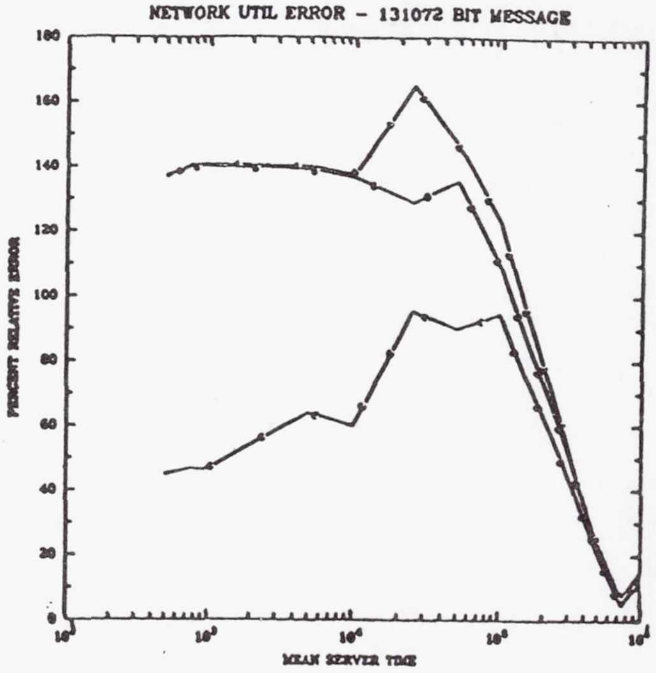


Figure 42.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1993	3. REPORT TYPE AND DATES COVERED Contractor Report

4. TITLE AND SUBTITLE
Advanced Launch System Multi-Path Redundant Avionics Architecture Analysis and Characterization

5. FUNDING NUMBERS
C NAS1-17964
WU 506-46-21-56

6. AUTHOR(S)
Robert L. Baker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Research Triangle Institute
P.O. Box 12194
Research Triangle Park, NC 27709-2194

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-4516

11. SUPPLEMENTARY NOTES
Technical Monitor: Felix L. Pitts
Final Report - Task 28

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 62

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The objective of the Multi-Path Redundant Avionics Suite (MPRAS) program is the development of a set of avionics architectural modules which will be applicable to the family of launch vehicles required to support the Advanced Launch System (ALS). To enable ALS cost/performance requirements to be met, the MPRAS must support autonomy, maintenance, and testability capabilities which exceed those present in conventional launch vehicles. The multi-path redundant or fault tolerance characteristics of the MPRAS are necessary to offset a reduction in avionics reliability due to the increased complexity needed to support these new cost reduction and performance capabilities and to meet avionics reliability requirements which will provide cost-effective reductions in overall ALS recurring costs.

A complex, real-time distributed computing system is needed to meet the ALS avionics system requirements. General Dynamics, Boeing Aerospace, and C.S. Draper Laboratory have proposed system architectures as candidates for the ALS MPRAS. The purpose of this document is to report the results of independent performance and reliability characterization and assessment analyses of each proposed candidate architecture and qualitative assessments of testability, maintainability, and fault tolerance mechanisms. These independent analyses were conducted as part of the MPRAS Part 2 program and were carried under NASA Langley Research Contract NAS1-17964, Task Assignment 28.

14. SUBJECT TERMS
ALS MPRAS Architectures, Digital Computers, Distributed Processors, System Assessment, Multi-Path Redundant Avionics Suite (MPRAS), Fault tolerance, Reliability, Redundancy

15. NUMBER OF PAGES
200

16. PRICE CODE
A09

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT